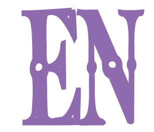
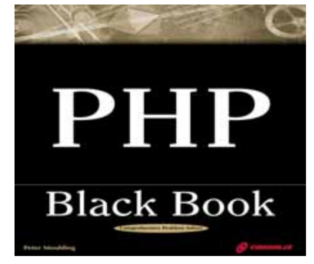
HACK X CRACK: ¿AUTENTIFICACIONES? ¿PASSWORDS? LO SENTIMOS, NO SIRVEN PARA NADA :)

PC PASO a PASO

APRENDE A PROGRAMAR EN







ESTE MES 16 páginas extra!!!

LOS CUADERNOS DE HACK X CRACK

www.hackxcrack.com

SERIE RAW

PROTOCOLO DNS



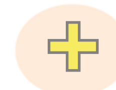
XML DOMNode

PROGRAMANDO EN

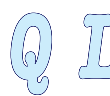


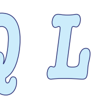
DESDE LINUX

VISUAL BASIC









SQL

3 SERVIDORES ON LINE PARA TUS PRACTICAS DE HACK

IP - HIJACKING

ENVENENAMIENTO DE DNS

SYN-ACK-FLAGS... ...

MANIPULANDO PAQUETES

DNS-TLD CACHE

JERARQUIA DNS

SUPLANTANDO SERVIDORES

SECUESTRO DE CONEXIONES

HACK X CRACK - HACK X CRACK - HACK X CRACK - HACK X CRACK

Nº 14 -- P.V.P. 4,5 EUROS



PC PASO A PASO: ¿CÓMO FUNCIONA INTERNET? DNS, EL ALMA DE LA RED!!!





EDITORIAL: EDITOTRANS S.L.
C.I.F: B43675701
PERE MARTELL Nº 20, 2º - 1ª
43001 TARRAGONA (ESPAÑA)

Director Editorial
I. SENTIS
E-mail contacto
director@editotrans.com
Título de la publicación
Los Cuadernos de HACK X CRACK.
Nombre Comercial de la publicación
PC PASO A PASO
Web: www.hackxcrack.com
Dirección: PERE MARTELL Nº 20, 2º - 1ª.
43001 TARRAGONA (ESPAÑA)



Director de la Publicación
J. Sentís

E-mail contacto
director@hackxcrack.com

Diseño gráfico:
J. M. Velasco

E-mail contacto:
grafico@hackxcrack.com

Redactores
AZIMUT, ROTEADO, FASTIC, MORDEA, FAUSTO, ENTROPIC, MEIDOR, HASHIMUIRA, BACKBONE, ZORTEMIUS, AK22, DORKAN, KMORK, MAILA, TITINA, SIMPSIM... ... ...

Contacto redactores
redactores@hackxcrack.com

Colaboradores
Mas de 130 personas: de España, de Brasil, de Argentina, de Francia, de Alemania, de Japón y algún Estadounidense.

E-mail contacto
colaboradores@hackxcrack.com

Imprime
I.G. PRINTONE S.A. Tel 91 808 50 15

DISTRIBUCIÓN:
SGEL, Avda. Valdeparra 29 (Pol. Ind.)
28018 ALCOBENDAS (MADRID)
Tel 91 657 69 00 FAX 91 657 69 28
WEB: www.sgel.es

TELÉFONO DE ATENCIÓN AL CLIENTE: 977 22 45 80
Petición de Números atrasados y Suscripciones (Srta. Genoveva)

HORARIO DE ATENCIÓN: DE 9:30 A 13:30
(LUNES A VIERNES)

NUMERO 14 -- PRINTED IN SPAIN
PERIOCIDAD MENSUAL
Deposito legal: B.26805-2002
Código EAN: 8414090202756

EDITORIAL

DICIEMBRE ... ...

DICIEMBRE… … un nuevo año está a punto de nacer y debemos seguir el ritual de comprar compulsivamente hasta liquidar el último céntimo de nuestra cuenta corriente y fulminar el saldo de nuestra tarjeta de crédito (para quien la tenga, claro).

Por primera vez, esta editorial no tratará sobre la revista, hay algo más importante, tu bolsillo. Como esta no es una revista que se vende al mejor postor vamos a darte un consejo que seguro ninguna otra publicación del sector informático se atreverá a dar :)

Estas navidades NO TE COMPRES UN ORDENADOR!!! Si quieres, date un caprichito, compra un poco de memoria, un monitor TFT de 18 pulgadas mínimo o haz un poco de "modding"… PERO NO ACTUALICES TU PLACA MADRE NI TU PROCESADOR.

No, no somos esquizofrénicos ni caemos en contradicciones. Cuando decimos que te des un capricho y te compres un TFT de 18" es porque ese monitor te durará 10 años, y si te gusta el "modding" pégale un vistazo al anuncio de la contraportada (si a nosotros nos ha sorprendido estoy seguro que no os dejará indiferentes). No, no me pagan para publicitar a nadie (seguro que ya estás pensando mal)… el mensaje que intento darte es QUE DISFRUTES de la informática, que no te dejes influir por esa enfermedad de "el procesador más rápido", "la tarjeta más espeluznante" y "la placa madre más capacitada", no, invierte en lo que te haga disfrutar…

¿Eres de los que disfruta teniendo el procesador más potente pero mantiene un triste y viejo monitor CRT de 17 pulgadas? Pues entonces escucha mi advertencia: El año 2004 traerá MUCHAS novedades dentro del campo informático, cambios de tecnología que dejarán atrás el bus AGP, el bus PCI y la tecnología de 32 bits… no tires el dinero actualizando tu equipo estas navidades!!!

Y FELICES FIESTAS!!!

IDICE DE ANUNCIANTES

IDICE DE ANUNCIANTES

CURSO DE PHP

EMPEZAMOS A PROGRAMAR PHP(I)

PHP es, con diferencia, la forma de programar páginas Web más extendida en Internet. Vamos a iniciar un nuevo curso que te permitirá empezar a utilizar este lenguaje y aportará a tus páginas una nueva dimensión.
Empezamos con un poco de historia, seguimos con la instalación de los elementos necesarios y nos vamos directamente a jugar con las variables. Que disfrutes!!!

Bienvenidos a un nuevo curso, a lo largo de los próximos meses aprenderemos a programar el lenguaje PHP. Son varias las razones que nos han impulsado a iniciar este curso ya que para comprender como funciona una web con algo de complejidad hay que saber como está programada internamente. Existen otros lenguajes de programación web pero PHP es el más extendido y conocido por todos, además es gratis. Este primer capitulo es muy básico, está orientado para lectores que empiezan por primera vez a programar y es necesario que rompan el hielo. En próximos capítulos veremos como programar socket, telnet remotos, ...

1. Un poco de culturilla no viene mal

Si piensas que PHP es nuevo estás equivocado, el nacimiento de PHP comenzó a gestarse en 1995 gracias a un producto llamada PHP/FI creado por Rasmus Lerdorf. Realmente este producto era un conjunto de scripts en Perl (es otro lenguaje de programación) para que Rasmus pueda controlar los accesos a su trabajo. Posteriormente lo hizo más potente rescribiendo todo el código en C y con nuevas funciones como acceso a base de datos. Contento con su trabajo tubo una gran idea "eligió liberar el código fuente de PHP/FI para que cualquiera pudiera utilizarlo".

En 1997 PHP ya estaba siendo utilizado por el 1% (50.000 dominios) de los dominios en Internet, aún el control de PHP/FI estaba en el creador. A finales de 1997 dos programadores llamados Andi Gutmans y Zeev Zuraski rescribieron por completo PHP/FI y lo llamaron PHP 3.0. Esta versión de PHP es bastante similar a lo que ya conocemos, hay pocas diferencias. En 1998 PHP ya estaba siendo utilizado por el 10% de los dominios y se liberó oficialmente en Junio de 1998, después de haber dedicado 9 meses a pruebas.

En invierno de 1998 ambos programadores crearon un nuevo motor PHP mucho más rápido y mejorando los accesos a las bases de datos además de ser bastante más estable ya que la anterior versión no estaba suficientemente preparada para aplicaciones complejas. Al nuevo motor lo llamaron Zend (comprimido de Zeev y Andi). La nueva versión fue bautizada como PHP 4.0 y el código se liberó en el año 2000. Actualmente se estima que PHP 4.0 esté siendo utilizada en el 20% de los dominios en Internet.

Para terminar, decir que ya se está trabajando en la nueva versión PHP 5.0, ¿qué nos deparará?, si la versión PHP 4.0 ya es alucinante. ¿cómo será PHP 5.0?, habrá que esperar un tiempo para saberlo.

2. ¿Qué se puede hacer con PHP?

Con PHP se pueden crear complejos programas, se utiliza mucho para el tratamiento de formularios, procesamiento de la información, mantener sesiones de usuarios (cookies y sesiones), en este curso aprendemos a utilizar

PHP de distintas formas, como script de servidor web y como script de aplicación, si no has entendido nada de lo comentado, no te preocupes ya se verá más adelante y con ejemplos. Para resumir podemos comentar que los scripts PHP son utilizados para:

- Scripts en la parte de servidor: es lo que estamos acostumbrados a ver en Internet, páginas que se ejecutan en el servidor web. Para ello se necesita un servidor web (Apache por ejemplo) y tener configurado el servidor web para que interprete las páginas PHP.
- Scripts en línea de comandos: esto no es tan conocido entre los programadores de PHP, se puede programar complejos scripts PHP y ejecutarlos sin necesidad de servidor web. ¿Qué utilidad puede tener programar un script en línea de comandos?, por ejemplo puedes crear un programa que borre la papelera de reciclaje cada 24 horas, ya lo veremos más adelante.
- Aplicaciones gráficas: Se puede programar aplicaciones gráficas al estilo de Visual Basic (con ventanas, botones, ...) pero para ello es necesario es necesario utilizar PHP-GTK.

En este curso trataremos de aprender a desarrollar bajo los dos primeros puntos, el desarrollo de aplicaciones básicas no lo aprenderemos, como mucho haremos algún ejemplo, pero sin dar mucha importancia, ya que existen otros lenguajes mejor preparados para estos menesteres como Visual Basic.

PHP puede ser utilizado en la mayor parte de sistemas operativos y servidores web: Linux, muchas variantes Unix (incluido HP-UX, Solaris y OpenBSD), Microsoft Windows, Mac OS X, RISC OS y probablemente alguno más. Soporta la mayoría de servidores web de hoy en día, incluyendo Apache, Microsoft Internet Information Server, Personal Web Server, Netscape y iPlanet, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd y muchos otros.

Pero no solo eso, además PHP puede trabajar con la mayoría de las bases de datos como: Adabas D, Ingres, Oracle (OCI7 y OCI8), dBase, InterBase, Ovrimos, Empress, FrontBase, PostgreSQL, FilePro (solo lectura), mSQL, Solid, Hyperwave, Direct MS-SQL, Sybase, IBM DB2, MySQL, Velocis, Informix, ODBC y Unix dbm.

Y lo más importante para este curso , también tiene soporte para comunicarse con otros servicios usando protocolos tales como LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (en Windows) y muchos otros. También se pueden crear raw sockets.

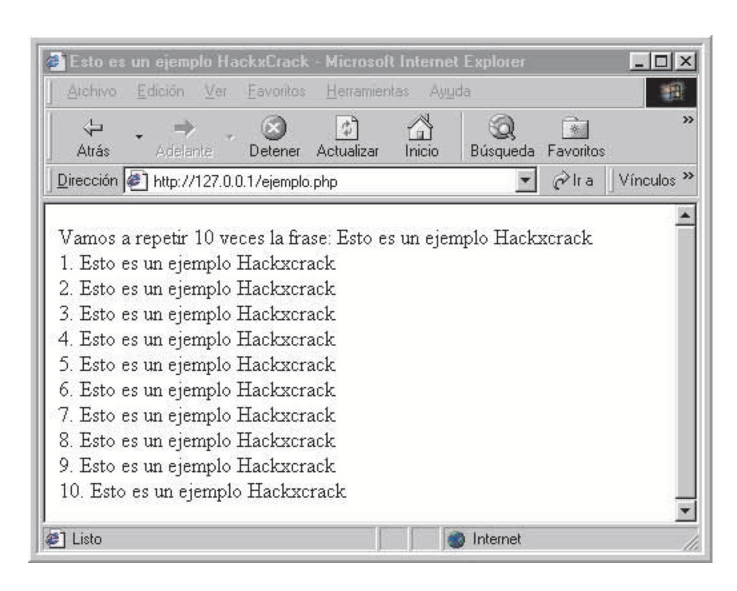
3. Pero qué es realmente PHP

PHP es un lenguaje interpretado de alto nivel embebido en páginas HTML y ejecutado en el servidor. ¿qué quiere decir todo esto?, muy sencillo, es un lenguaje que no necesita ser compilado. Si estás siguiendo el curso de Visual Basic estarás viendo que para ejecutar el programa primero tienes que compilarlo y luego ejecutarlo. Un lenguaje interpretado no necesita la compilación, es decir, puedes crear un programa y ejecutarlo directamente. Lógicamente esto tiene sus ventajas y problemas. Las ventajas es que puede ser considerado más rápido a la hora de programar y de depurar posibles errores, como problema podemos comentar que un lenguaje interpretado es más lento ya que el código necesita ser interpretado y luego ser ejecutado. Otra desventaja es que el código puede ser visible por cualquier persona con permisos para ello. En la actualidad la velocidad no es problema, ya que cada vez los ordenadores son más rápidos y se puede considerar un obstáculo para programar lenguajes interpretados siempre y cuando la velocidad extrema no sea crucial.

PHP es un lenguaje embebido en HTML, esto quiere decir que puedes programar tus páginas HTML e introducir código PHP, este código PHP

no se mostrará al navegante ya que es ejecutado en el servidor. Lo mejor es un ejemplo para comprender todo lo comentado.

Si queremos crear una página web que muestre 10 veces la frase "Esto es un ejemplo Hackxcrack" podemos hacerlo mediante HTML sin problemas pero tambien podemos hacerlo con PHP y de manera más sencilla, ya que gracias al PHP podemos pasar por URL el número de veces que aparezca la frase. El código PHP de este sencillo ejemplo sería:




```
<html>
<head>
<title>Esto es un ejemplo HackxCrack</title>
</head>
<body>
Vamos a repetir 10 veces la frase: Esto es un ejemplo
Hackxcrack<br>
<? for
($contador=1;$contador<=10;$contador=$cont
ador+1) { ?>
<? print $contador; ?>. Esto es un ejemplo
Hackxcrack<br>
<? } ?>
</body>
</html>
```

Analizando el código se puede ver código HTML y código PHP, lo que hace este código PHP es un bucle de 10 repeticiones (todo esto ya lo explicaremos más adelante). Es importante saber que el código PHP es ejecutado en el servidor y que este código no es mostrado en el cliente, si el navegante decidiera ver el código de la página mostrada en su navegador simplemente vería 10 veces la frase en perfecto HTML pero nada de PHP.

4. Instalar PHP

¿Recuerdas que en el número 10 de hackxcrack se explicaba como instalar el PHP en Apache?, busca en tu estantería la revista nº10 y encontrarás un artículo llamado "Apache Parte IV Trio de Ases Apache-PHP-Mysql", te recomendamos que leas y sigas los pasos comentados del nº10, ya que lo necesitarás para comenzar con las prácticas de PHP.

En caso de que no tengas la revista tienes dos opciones, pedir que te envíen el número 10 (es lo que te recomiendo) o utilizar uno de los paquetes existentes en Internet que te instalan el Apache, el PHP y el MySQL. Además te lo dejan todo 100% operativo. ¿dónde puedes encontrar uno de estos paquetes?, hay muchos. Puedes bajarte el Appserv, es muy sencillo de instalar (simplemente hay que ejecutar un programa y esperar a que se instale todo). Puedes encontrarlo en:

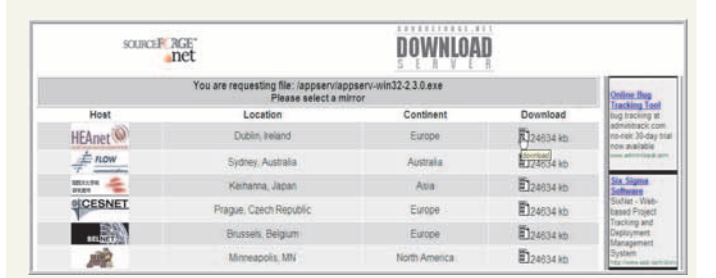
http://www.appservnetwork.com



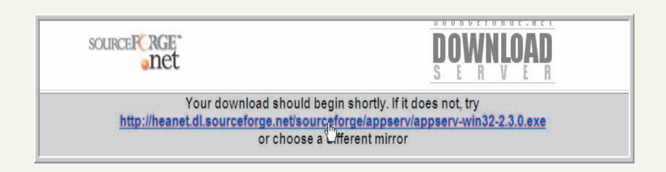
Descarga de APPSERV:
1.- Abre tu navegador y dirígete a
http://www.appservnetwork.com
2.- Una vez visualices la página pulsa sobre el enlace que hay debajo de "DOWLOAD APPSERV 2.3.0"



3.- Llegarás a una página donde tienes a tu disposición varios servidores desde donde descargar el archivo, simplemente pulsa sobre uno de ellos, bajo la columna download. Nosotros lo descargamos desde el primero de ellos, puedes verlo en la imagen.



4.- Cuando pulses se abrirá otra ventana y empezará la descarga. Si la descarga no empieza de forma automática pulsa sobre el enlace que aparece al principio de la página, tal y como puedes ver en la captura.



5.- Una vez empiece la descarga del archivo, el navegador te preguntará dónde quieres guardarlo, pues eso, guárdalo donde quieras, por defecto en el disco C:

Si, seguro que ves del todo innecesaria esta explicación, pero la experiencia manda y recibimos muchos mails preguntando cosas tan simples como esta :)

Instalando el APPSERV (tampoco deberíamos explicarlo...pero bueno…):

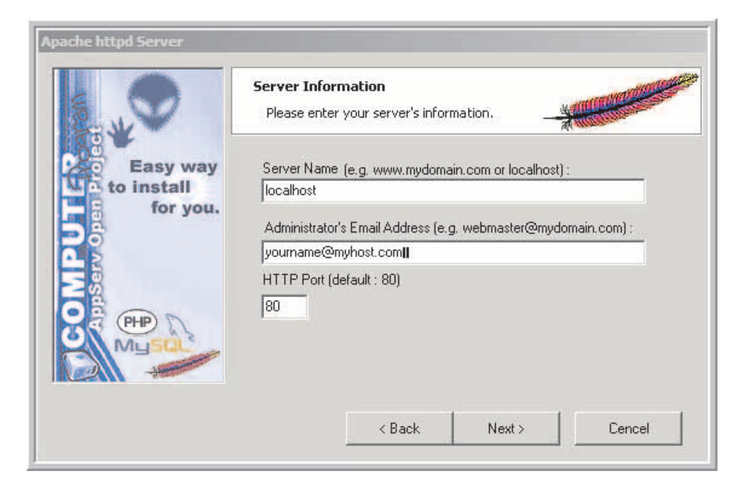
1.- Ejecuta el archivo "appserv-win32-2.3.0.exe", que debería estar en el disco duro C: si seguiste las instrucciones de descarga.

2.- Aparecerá la típica ventanita de instalación, pulsa sobre el botón NEXT

3.- Ahora aparecerá otra ventanita preguntándote dónde quieres instalar el programa. Fíjate que por defecto lo instalará en "C:\AppServ", pues muy bien, le decimos que si pulsando sobre el botón NEXT.

4.- Ahora se abrirá otra ventanita preguntando el tipo de instalación que deseamos, pues nos conformamos con la "typical" pulsando el botón NEXT :)

5.- Ahora se abrirá otra ventanita que para cualquier lector habitual de PC PASO A PASO no tiene ninguna dificultad, pero si nunca instalaste APACHE tal y como te enseñamos en los números anteriores, quizás no sepas qué hacer. La ventana es la siguiente:



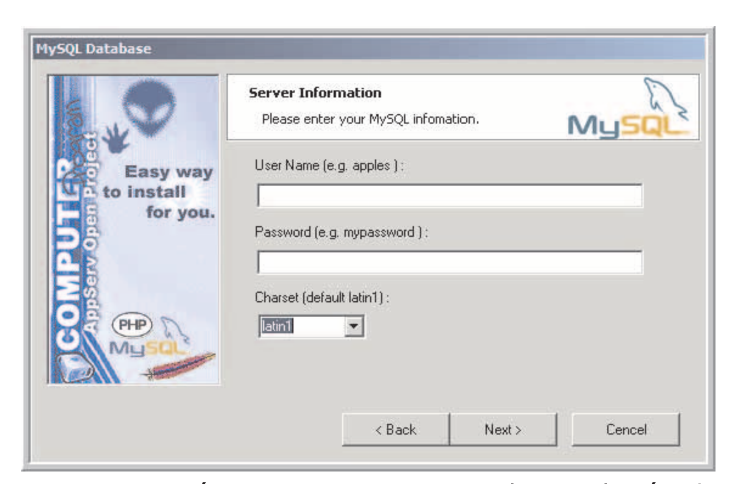
SERVER NAME… ¿qué es eso?
Bueno, bueno, bueno… si te haces esa pregunta eres de los que no nos ha leído nunca ;p Vale, muy rapidito… en principio, estás instalando un Servidor Web en tu ordenador para que cualquier persona de Internet pueda acceder a él y ver

una página Web creada por ti; pero en este caso NO QUEREMOS que eso suceda, solo quieres practicar los ejercicios de PHP ¿VERDAD? Pues entonces pondremos una palabra muy rara llamada "localhost" (tal y como puedes ver en la imagen. Haciendo esto, SOLO TU y NADIE MAS QUE TU podrá acceder al servidor, puesto que "localhost" significa que tu Servidor Apache será accesible en la IP 127.0.0.1, una IP LOCAL a la que solo tu equipo tiene acceso (para más información repasa los números anteriores).

ADMINISTRATOR´S E-MAIL... ¿qué pongo? Puedes poner un mail verdadero o un mail falso (por ejemplo soydios@soymuybueno.com), no importa, para más "info" repasa los números anteriores o pregunta en el foro de PC PASO A PASO (www.hackxcrack.com).

HTTP PORT DEFAULT... ¿?
Esto ya se ha explicado mil veces, déjalo en el 80 y así nos quitamos complicaciones. A partir de ahora cuando quieras acceder a tu servidor web deberás abrir el navegador (Internet Explorer, Netscape... ...) e ir a la dirección www.localhost.com o http://127.0.0.1 (es exactamente lo mismo)

6.- Pulsamos next :p y nos aparecerá otra ventanita.

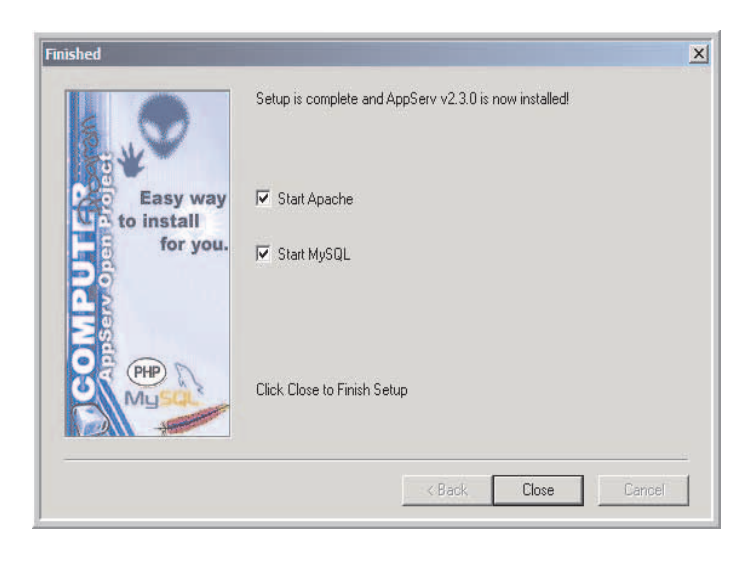


Fíjate MUY BIEN en que la instalación de APACHE ya ha "finalizado", ahora estamos instalando MySQL y que se nos está pidiendo un USER NAME (nombre de usuario) y un PASSWORD, pues venga, nosotros pondremos de nombre de usuario DOMINUS1 y de password DEMONIO1.

No cometas "el error del novato" (y te aseguro que todos lo hemos hecho muchas, muchas, muchas veces) de no apuntar el nombre de usuario y el password en un papelito. APÚNTATE EL NOMBRE DE USUARIO Y EL PASSWORD y recuerda que está en MAYUSCULAS, lo necesitarás cuando tengas que acceder a la base de datos (MySQL).

El CHARSET lo dejamos por defecto como "latin1" y pulsamos NEXT.

7.- LISTO!!!... a esperar que se instale y salga una ventanita donde pulsaremos el botón CLOSE.



Fíjate que están marcados "Start Apache" y "Start MySQL"... pero dijimos que también estábamos instalando el PHP ¿verdad? Bueno, pues gracias al APPSERV, el PHP se integra de forma totalmente automática en el Servidor Apache :) (como ya hemos dicho, si quieres saber más repasa los números anteriores).

En este momento tendrás el Servidor APACHE instalado y ejecutándose, para comprobarlo tan solo tienes que abrir tu explorador y poner

http://localhost (o http://127.0.0.1 **-es exactamente lo mismo-**) y pulsar enter :p, podrás ver la siguiente pantallita:



Ya tienes el APACHE configurado para poder ejecutar código en PHP e incluso utilizar la base de datos MySQL. Fíjate que debes tener un nuevo icono junto al reloj del sistema, es una especie de semáforo, pues ese es el administrador de MySQL, dejémoslo ahí por el momento.

Ya está, ya estamos preparados!!! Y si estás pensando que toda esta explicación sobre la instalación sobraba, tienes razón... pero claro, después a ver quien es el valiente que responde los mails que nos llegan :)



En los números...

En números anteriores de PC PASO A PASO aprendimos a instalar y configurar el servidor APACHE, también vimos la instalación de PHP y MySQL y configuramos APACHE para poder operar con ellos.

Para cuando leas estas páginas intentaremos tener en la WEB los artículos en que tratamos este tema para que puedas descargarlos gratuitamente. De todas maneras, PC PASO A PASO es una especie de curso continuo donde todo lo que aprendes en un número anterior tarde o temprano es utilizado en números posteriores. Es muy recomendable que no te pierdas ningún número!!! ;)

5. El primer programa

Ya está todo instalado, ¿verdad?, ahora vamos a crear un programa para que nos muestre en una página web el famoso mensaje de "Hola mundo".

Para programar puedes utilizar el bloc de notas, simplemente crea un fichero con extensión PHP y llámalo ejemplo1, es decir, crea un fichero llamado ejemplo1.php

Recuerda que este fichero tienes que colocarlo en el directorio raíz del servidor web, si has instalado el Appserv la ruta es: c:\appserv\www\

Pon el siguiente código:

```
<html>
<head>
<title>Ejemplo Hola Mundo</title>
</head>
<body>
<h1><? echo "HOLA MUNDO"; ?></h1>
</body>
</html>
```

Para probarlo pon la url http://127.0.0.1/ejemplo1.php y verás el resultado. Mira el código y observarás que el código PHP no se encuentra por ningún lado :) (Si pones http://localhost/ejemplo1.php no dudes que también funcionará).

NOTA: ¿no sabes cómo ver el código de una página Web? Hay que ver... Bueno, vale... Si tienes el Internet Explorer 6, tan solo tienes que ir a una página cualquiera (por ejemplo www.google.com) y una vez visualizada ir al menu "Ver" y pulsar sobre la opción "Código Fuente"... listo!!!

Este ejemplo es demasiado sencillo, pero muy útil para explicar los tags de PHP, el código PHP siempre tiene que ir entre <? ... ?>, todo lo que aparezca entre estos símbolos será

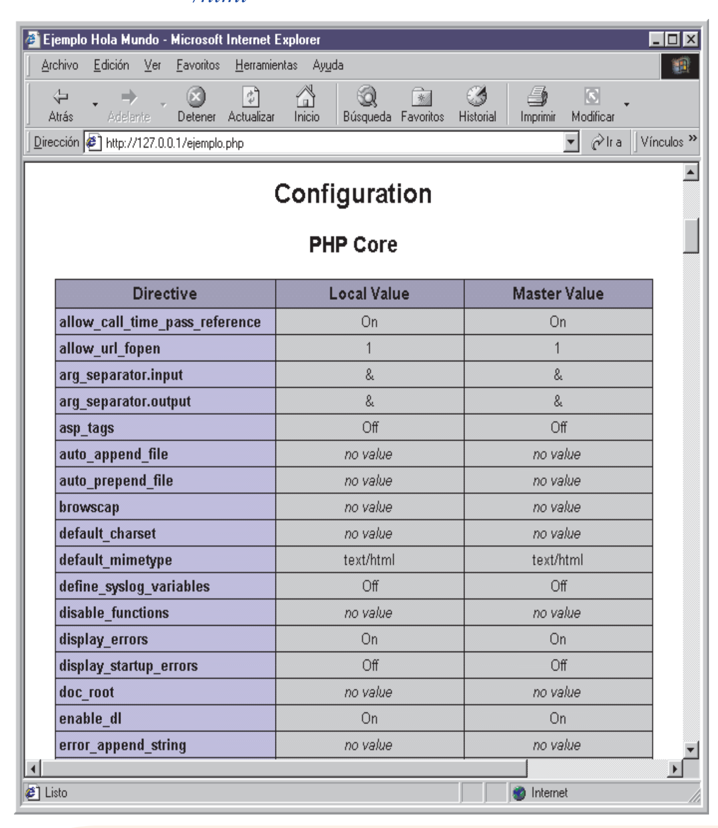
interpretado como PHP y ejecutado. En el ejemplo anterior hemos mostrado el mensaje utilizando el comando ECHO de PHP.

Las sentencias en PHP finalizan con un punto y coma, no olvides esto pues es un error común no finalizar una línea de código con el punto y coma.

Es interesante conocer la configuración de PHP, phpinfo() nos proporciona información de gran valor como: el sistema operativo, servidor web, path donde se encuentra el fichero de configuración, librerias instaladas, ...

Otro ejemplo, veamos como está configurado PHP, para ello creamos un nuevo programa aún más simple: (igual que antes, escribe el código en el block de notas y guardalo con elnombrequequieras.php en el directorio c:\appserv\www\, después llámalo desde el navegador escribiendo http://127.0.0.1/elnombrequequieras.php)

```
</head>
<body>
<? phpinfo(); ?>
</body>
</html>
```



Directive	Local Value	Master Value
allow_call_time_pass_reference	On	On
allow_url_fopen	1	1
arg_separator.input	&	&
arg_separator.output	&	&
asp_tags	Off	Off
auto_append_file	no value	no value
auto_prepend_file	no value	no value
browscap	no value	no value
default_charset	no value	no value
default_mimetype	text/html	text/html
define_syslog_variables	Off	Off
disable_functions	no value	no value
display_errors	On	On
display_startup_errors	Off	Off
doc_root	no value	no value
enable_dl	On	On
error_append_string	no value	no value

6. Empezamos de verdad con PHP y a toda velocidad

6.1 Variables.

En PHP las variables se representan como un signo de dólar seguido por el nombre de la variable. El nombre de la variable es sensible a minúsculas y mayúsculas. Un nombre de variable válido tiene que empezar por una letra o el signo de raya, ejemplos:

```
<?
$nombre = "Juan";
$Nombre = "Antonio";
echo "$nombre, $Nombre";    // Mostrará: "Juan, Antonio"
$101nombre = 'Pedro';   // Incorrecto ya que impieza con un número.
$_101nombre = 'Pedro';   // Correcto ya que empieza con una raya.
?>
```

Los enteros se puede especificar utilizando cualquiera de las siguientes sintaxis:

```
$a = 1234; # número decimal
$a = -123; # un número negativo
$a = 0123; # número octal (equivalente al 83 decimal)
$a = 0x12; # número hexadecimal (equivalente al 18 decimal)
```

Importante: Las variables comienzan con el signo del dólar. Ejemplo de variables:

```
<?
$a=100;
$b=5;
$c=$a;
$total=$a-$b;
print $total; // Resultado 95
¿>
```

En PHP las variables se asignan por valor, ¿qué quiere decir esto?, pues si miramos el ejemplo anterior vemos que la variable $c ha tomado el valor de la variable $a y que el valor de la variable $a no ha cambiado, es decir sigue valiendo 100.

En PHP también se puede asignar variables por referencia y para ello hay que utilizar el signo &, pongamos un ejemplo para comprender a que se refiere por referencia:

```
<?
$var1 = 'hackxcrack';
$var2 = &$var1;
$var2 = "La mejor revista $var1";
echo $var1;
echo $var2;
?>
```

Según el ejemplo, la variable $var1 toma el valor "hacxcrack" y la variable $var2 es creada por referencia apuntando a la variable $var1 (fijaros que se ha colocado el signo &). Cualquier cambio que se haga en $var2 afectará a la variable $var1, según este ejemplo la variable $var2 cambia de valor pero también lo hace la variable $var1. El resultado es que tanto la variable $var1 y $var2 toman el valor "La mejor revista hackxcrack".

Variables de variables

Tal vez lo que vamos a explicar no tengas que utilizarlo pero es conveniente saber que PHP acepta tener nombre de variables variables. No es juego de palabras, lo que quiere decir es que son nombre de variables que se pueden usar dinámicamente.
Una variable normal se asigna de la siguiente manera:

```
<?
$a="hola";
$$a="mundo";
echo "$a $hola"; // mostrará hola mundo
¿>
```

Una variable variable tomará el valor de la variable $a y lo convertirá en otra variables, para ello se utiliza dos signos $.

Con estos ejemplos hemos creado dos variables, una variable $a y otra variable llamada $hola.

Capturando variables externas

PHP puede capturar las variables enviadas por un formulario HTML para su posterior manejo, como ya sabéis un formulario HTML puede enviar los datos de dos maneras (POST y GET), PHP dispone de diferentes formas para capturar los datos.
Para capturar una variable POST:

```
$HTTP_POST_VARS['username'];
```

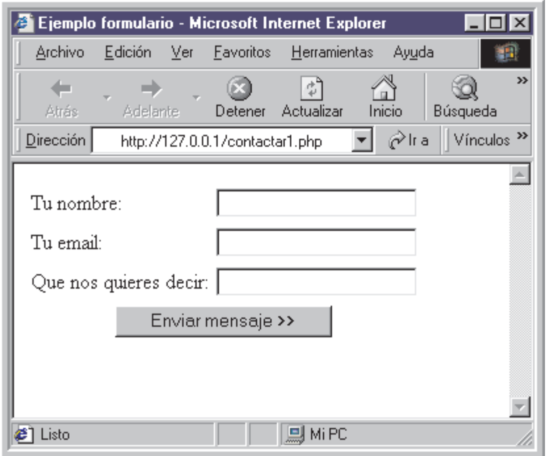
Para capturar una variable GET:

```
$HTTP_GET_VARS['username'];
```

Pongamos un ejemplo sencillo, hagamos el típico programa de "contactar", en donde el navegante desea enviar un mensaje al webmaster y para ello rellena un formulario. Los datos serán enviados por email a webmaster@tudominio.com.

```
<? if ($HTTP_POST_VARS['email']!="") {
$datos="Nombre:$HTTP_POST_VARS['$nombre']\r\n";
$datos=$datos."Tu email:$HTTP_POST_VARS['$email']\r\n";
$datos=$datos."Mensaje:$HTTP_POST_VARS['$mensaje']";
mail("webmaster@tudominio.com","Informacion de
contacto",$datos);
}
?>
<html>
<head>
<title>Ejemplo formulario</title>
<meta http-equiv="Content-Type" content="text/html; charset=iso-8859-1">
</head>
<body bgcolor="#FFFFFF" text="#000000">
<form name="form1" method="post" action="contactar.php">
  <table width="300" border="0" cellspacing="0" cellpadding="3">
    <tr>
      <td width="49%">Tu nombre:</td>
      <td width="51%">
        <input type="text" name="nombre">
      </td>
    </tr>
    <tr>
      <td width="49%">Tu email:</td>
      <td width="51%">
        <input type="text" name="email">
      </td>
    </tr>
    <tr>
      <td width="49%">Que nos quieres decir:</td>
      <td width="51%">
        <input type="text" name="mensaje">
      </td>
```

```
    </tr>
    <tr align="center">
      <td colspan="2">
        <input type="submit" name="Submit" value="Enviar mensaje &gt;&gt;">
      </td>
    </tr>
  </table>
</form>
</body>
</html>
```



Este pequeño pero eficaz programa muestra un formulario solicitando 3 datos (nombre, email y mensaje), los datos son enviados a la misma página y se ejecuta el código PHP en el servidor, PHP recupera los datos enviados mediante $HTTP_POST_VARS y utiliza la variable $datos pare componer un mensaje, posteriormente el mensaje es enviado a la dirección webmaster@tudominio.com. En próximos capítulos hablaremos de la función mail(), aunque de esta forma tan sencillo has aprendido a enviar emails utilizando el lenguaje PHP.

Constantes

Se definen como constantes los valores definidos y que su valor no va a ser modificado a lo largo de la ejecución de la página PHP. Por ejemplo, el número PI es una constante y para definirlo en PHP como constante habría que hacer lo siguiente:

```
<?
Define("pi",3.141592);
Print pi;
¿>
```

Observa como en este caso no es necesario utilizar el símbolo $ delante del nombre de la constante, ya que no se trata realmente de una variable si no de una constante.

Las diferencias entre constantes y variables :

- Las constantes no son precedidas por un símbolo de dolar ($)
- Las constantes solo pueden ser definidas usando la función() define , nunca por simple asignación
- Las constantes pueden ser definidas y accedidas sin tener en cuenta las reglas de alcance del ámbito.
- Las constantes no pueden ser redefinidas o eliminadas después de establecerse; y
- Las constantes solo puede albergar valores escalares

Conversión de variables

En PHP podemos convertir variables de un tipo en otro, para ello se antepone el tipo de dato que se quiere obtener, por ejemplo:

```
<?
$variable1=66.6;
$variable2= (int) $var1;
print $variable2; // mostrará en pantalla 66
¿>
```

Funciones de variables

Gettype(), Esta función devuelve el tipo de dato, es utilizado para conocer el tipo de dato de una variable. Ejemplo:

```
<?
$color="Amarillo";
$pi=3.14:
print (gettype($color)); // muestra en pantalla String
print (gettype($pi)); // muestra en pantalla double
¿>
```

Settype(), Establece el tipo de dato que va a guardar una variable. Esta función no es

necesaria utilizarla cada vez que deseemos crear una variable, aunque no estaría mal utilizarla para evitar posibles errores ya que cuando los códigos son grandes uno ya no sabe que tipos de variables está manejando. Ejemplo:

```
<?
$color=”blanco”;
settype($color,”string”);
¿>
```

Isset(), se utiliza para determinar si una variable ha sido iniciada con un valor, si ha sido asignada devuelve true.

```
<?
$pais=”España”;
if (isset($pais)) { print “Ha sido asignada”;} else { print “No ha sido asignada”;}
¿>
```

Unset(), es utilizada para destruir variables y por lo tanto para liberar recursos. No olvides utilizar esta variable para liberar memoria, es una función muy recomendada y es poco utilizada (tal vez por desconocimiento).

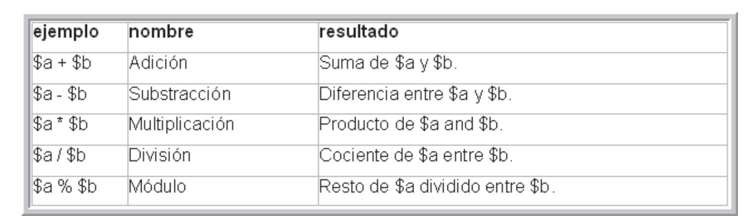
```
<?
$colores=”amarillo, verde, rojo”;
unset($colores); // se ha destruido la variable $colores.
>?
```

7. Operadores

Primero vamos a nombrar los 4 principales operadores y luego haremos un ejemplo que muestre todos los operadores.

7.1 Operadores aritméticos

Con en la gran mayoría de lenguajes de programación, en PHP existen cinco operadores atirméticos:



ejemplo	nombre	resultado
$a + $b	Adición	Suma de $a y $b.
$a - $b	Substracción	Diferencia entre $a y $b.
$a * $b	Multiplicación	Producto de $a and $b.
$a / $b	División	Cociente de $a entre $b.
$a % $b	Módulo	Resto de $a dividido entre $b.

7.2 Operadores de comparación

Como su nombre indica son operadores que nos permiten comparar dos variables.

ejemplo	nombre	resultado
$a == $b	Igualdad	Cierto si $a es igual a $b.
$a === $b	Identidad	Cierto si $a es igual a $b y si son del mismo tipo
$a != $b	Desigualdad	Cierto si $a no es igual a $b.
$a < $b	Menor que	Cierto si $a es estrictamente menor que $b.
$a > $b	Mayor que	Cierto si $a es estrictamente mayor que $b.
$a <= $b	Menor o igual que	Cierto si $a es menor o igual que $b.
$a >= $b	Mayor o igual que	Cierto si $a es mayor o igual que $b.

7.3 Operadores lógicos

Son utilizados para combinar varias condiciones.

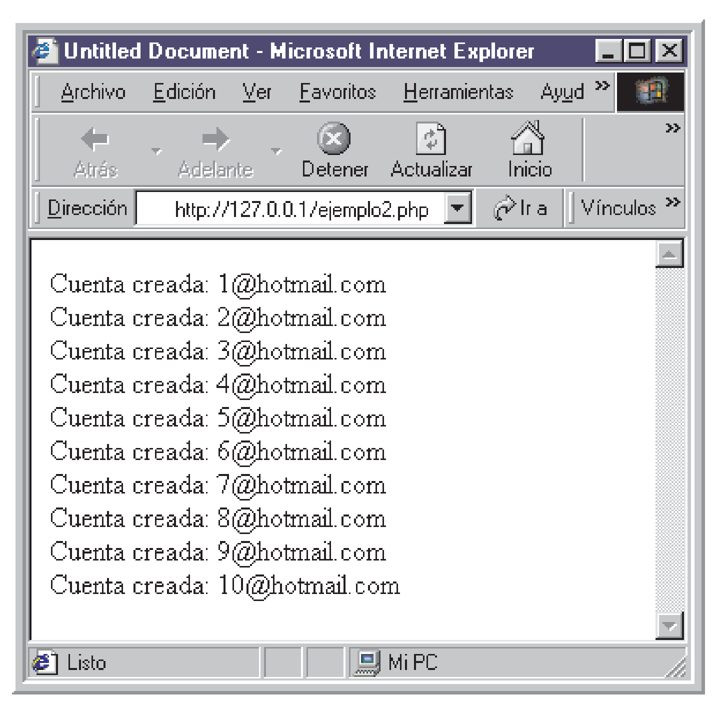
ejemplo	nombre	resultado
$a and $b	Y	Cierto si tanto $a como $b son ciertos.
$a or $b	O	Cierto si $a o $b son ciertos.
$a xor $b	O exclusiva	Cierto si $a es cierto o $b es cierto, pero no ambos a la vez.
! $a	Negación	Cierto si $a no es cierto.
$a && $b	Y	Cierto si tanto $a como $b son ciertos.
$a || $b	O	Cierto si $a o $b son ciertos.

Ejemplo utilizando operadores:

Vamos a crear un sencillo programa que genere 10 cuentas de correo de un dominio y que les envíe un correo de saludo, además los usuarios de los mails serán números, es decir, 1@hotmail.com, 2@hotmail.com, ... La verdad es que es un ejemplo un poco tonto, pero seguro que muchos de vosotros estaréis viendo otras aplicaciones, ¿verdad?

El código sería:

```
<?
For ($contador=1;$contador<=10;$contador++) {
$cuenta=$contador.”@hotmail.com”;
Print “Cuenta creada: $cuenta <br>”;
mail($cuenta,"Saludos",”Saludos”);
}
¿>
```



En el ejemplo podemos ver la instrucción for, lo veremos más adelante pero lo importante es fijarnos en el interior de los paréntesis de la instrucción for, muestra:

Asignación: $contador=1
Operador de comparación: $contador<=10
Operador de incremento: $contador++

En el próximo número ...

Ya hemos roto el hielo, en el próximo número avanzaremos mucho más rápido. Mientras podéis practicar con vuestros primeros programas en PHP, hacer pruebas y más pruebas, que muy pronto comenzaremos a programar sockets.

David C.M

SERIE RAW (8)
DNS (DOMAIN NAME SYSTEM)

Todo llega, y en esta ocasión le ha tocado el turno al PROTOCOLO DNS. Por primera vez en la Serie RAW vamos a ver un protocola basado en UDP :) El sistema DNS es el alma de Internet, aprende con nosotros su funcionamiento!!!

1. Introducción

Un mes más, nos enfrentamos a los protocolos más importantes de la red, pero en este caso se trata de un protocolo muy especial. Y afirmo esto principalmente por dos motivos. En primer lugar, porque es un protocolo que utilizamos constantemente (cada vez que visitamos una página web, cada vez que escribimos un e-mail, cada vez que conectamos con un FTP, cada vez que hacemos Telnet a una máquina, etc, etc). Y en segundo lugar porque, por primera vez en la ya veterana serie RAW, no se trata de un protocolo basado en TCP, si no en el aún desconocido **UDP**!

En realidad el protocolo DNS puede funcionar tanto con TCP/IP como con UDP/IP, pero en la práctica el UDP es el que se utiliza, salvo para ciertas circunstancias especiales que ya veremos más adelante.

En vista de la novedad que esto supone, me veo obligado a comenzar el artículo explicando brevemente en que consiste el protocolo UDP, para luego pasar a los detalles del sistema y el protocolo DNS, y por último terminaré explicando lo devastadores que pueden llegar a ser los ataques contra el sistema DNS. ¡Allá vamos!

2. El protocolo UDP/IP vs TCP/IP

No es plan ahora de explicar detalladamente en qué consisten estos dos protocolos, ya que esto sería tema no para otro artículo, si no para una serie completa, así que nos conformaremos con pillar sólo un poco la idea. Como ya sabemos, todos los datos en **TCP/IP** circulan a través de una **conexión** entre un cliente y un servidor. Los datos podrán circular siempre que se mantenga activa esa conexión, y si ésta se cierra habrá que establecer una nueva conexión si se desea continuar la comunicación. Esto significa que el protocolo TCP/IP es un protocolo **orientado a conexión**.

Pero si lo pensamos detenidamente, nos daremos cuenta de que en realidad no se establece nunca ninguna conexión física entre el cliente y el servidor. En Internet no hay unas simpáticas telefonistas recibiendo todas las peticiones de conexiones en espera de enchufar el cable que une el cliente con el servidor, si no que en realidad todo está conectado con todo. Por tanto, las conexiones que se establecen son virtuales.

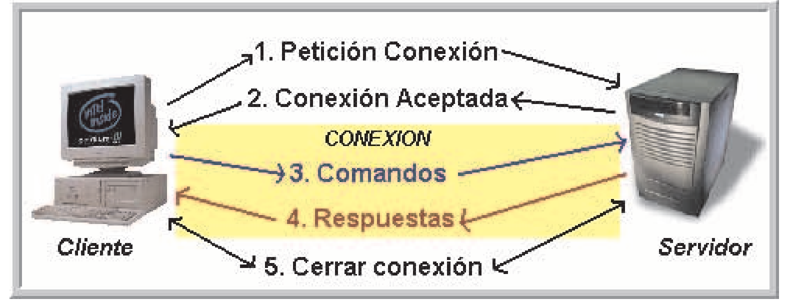


Fig 1.- Esquema del protocolo TCP/IP, orientado a conexión

En cambio, en el protocolo **UDP/IP**, no es necesario establecer una conexión virtual para permitir la comunicación, ya que se trata de un protocolo **no orientado a conexión**. En este

caso, el cliente envía los datos directamente a la IP del servidor, sin ninguna seguridad de si éste escuchará su petición, o incluso de si éste existe. A costa de esta escasa fiabilidad, el sistema es más rápido y sencillo que en el caso de un protocolo orientado a conexión.

Fig 2.- Esquema del protocolo UDP/IP, no orientado a conexión

En el caso que nos ocupa, que es el del protocolo **DNS**, que suele utilizar el protocolo UDP para el transporte de los datos (igual que FTP, HTTP, y todos los que hemos visto hasta ahora utilizaban TCP para el transporte de los datos), podemos tener cierta certeza de que el servidor ha escuchado nuestra petición, ya que éste responderá con otro bloque de datos en el que nos dará la respuesta a nuestra solicitud.

El que ya no sabrá si esa respuesta nos ha llegado o no es el propio servidor, ya que nosotros no le enviaremos ninguna confirmación de que la hemos recibido, pero eso ya es problema nuestro, y en caso de que no nos llegase la respuesta probablemente reintentaríamos la solicitud.

3. EL SISTEMA DNS

Entramos ya de lleno en el asunto, y empezaremos con una breve introducción (aunque espero que la mayoría sepáis ya en qué consiste el DNS), para luego pasar a detallar la arquitectura y el funcionamiento del protocolo.

Por supuesto, antes de nada os digo cuales son los RFCs que definen el protocolo, que son el **RFC 1034**, y el **RFC 1035** (ftp://ftp.rfc-editor.org/in-notes/rfc1034.txt y ftp://ftp.rfc-editor.org/in-notes/rfc1035.txt)

Como ya sabemos, cualquier máquina conectada de forma directa a Internet tiene una dirección única que la identifica.

Estas direcciones consisten en un número de **32 bits**, que se suele representar mediante 4 dígitos separados por puntos. Cada uno de estos dígitos está en base 256, es decir, están comprendidos entre 0 y 255.

Por supuesto, estoy hablando de las **direcciones IP**. Con 32 bits se pueden direccionar más de 4 mil millones de máquinas.

Quizá esto nos pueda parecer mucho, pero en realidad esta cifra se está convirtiendo en un problema, ya que se está quedando insuficiente para el crecimiento exponencial de la red, por lo que éste es uno de los motivos por el que se está implantando el sistema **IPv6** que sustituirá al actual protocolo IPv4 (y también a otros tantos relacionados directamente), y que utilizará direcciones de **128 bits**, lo cual permitirá direccionar nada menos que varios sixtillones de máquinas, lo cual es de esperar que sea suficiente mientras la raza humana siga confinada en el planeta Tierra.

Todo esto os lo cuento para que penséis en el terrible problema que sería para cualquier persona tener que acordarse de todos esos números, o bien tener que consultarlos cada vez que se quisiera acceder a una máquina, como si de una guía telefónica se tratase (de hecho, este sistema de "guía telefónica" fue el utilizado en los comienzos de la red Arpanet, ahora convertida en Internet, en la cual existía un único fichero **HOSTS.TXT** que contenía todas las direcciones de todas las máquinas de la red). A los seres humanos, en general, no se nos dan muy bien los números, ya que preferimos utilizar las palabras. De ahí surgió la necesidad de realizar un sistema de traducción de todos esos numerajos a palabras más intuitivas y más cómodas para el uso de los humanos. Gracias a este sistema, podemos

escribir en nuestro navegador www.google.com en lugar de escribir 216.239.39.99 o, lo que sería peor aún:
11011000111011110010011101100011
(que es la dirección IP de Google en binario).

Además, este sistema de traducción de IPs a nombres nos permite también no tener que depender de los cambios de ubicación de los servidores. Si, por ejemplo, la IP de Google cambiase, nosotros no tendríamos ni que saberlo, ya que su nombre seguiría siendo el mismo independientemente de la IP, y sería problema del sistema DNS el apañárselas para que la nueva IP estuviese asociada a www.google.com.

3.1. Arquitectura del sistema DNS

Esto que en principio puede parecer relativamente sencillo, no lo es tanto, teniendo en cuenta que se trata de traducir miles de millones de direcciones que pueden estar apareciendo, desapareciendo, o cambiando constantemente.

Y el resultado de estas traducciones tiene que estar accesible para todos y cada uno de los cientos de millones de usuarios de la red. Por tanto, este sistema requiere ante todo una arquitectura inteligente que permita llevar a cabo esta labor sin dar lugar a congestiones o cualquier otro tipo de problemas.

Empecemos imaginando el caso más sencillo: un único servidor DNS en Internet que mantuviese el registro de la base de datos de traducciones de direcciones IP a nombres. Cada vez que visitamos una página web, cada vez que escribimos un email, cada vez que conectamos con el IRC, cada vez que entramos en un FTP y, en resumen, cada vez que hacemos prácticamente cualquier cosa en Internet, necesitamos realizar una traducción de nombres a IPs. Esto ocurre igual para cada uno de los millones de usuarios de Internet. Esto supondría que cada usuario lanzaría varias peticiones de traducción por minuto, lo cual, multiplicado por el número de usuarios de la red, supondría miles de millones de conexiones por minuto. Por supuesto, esto es totalmente imposible de mantener para cualquier servidor. Y ni que decir tiene del problema que surgiría si éste servidor se cayese, se estropease, o tuviese cualquier otro problema...

Podemos pensar en una primera solución que suavizaría bastante las cosas, y es que las máquinas de cada cliente fuesen almacenando en su propia memoria las traducciones ya solicitadas previamente, para no tener que solicitar una traducción cada vez que se quisiese realizar una conexión con la misma máquina. Así surge el concepto de **cache de direcciones**, que es un sistema que en efecto se utiliza, aunque de forma bastante más compleja, ya que se realiza a muchos niveles, y no sólo en la máquina del cliente. Pero analicemos la nueva situación. Supongamos que, en el mejor de los casos, el 80% de las conexiones fuesen con máquinas cuya IP ya se tradujo anteriormente, por lo que no habría que volver a solicitar la traducción. En ese caso, aún ese 20% de conexiones seguirían generando un tráfico de miles de millones de conexiones, imposible de soportar por ningún servidor.

Por tanto, la única solución consiste en **descentralizar** el sistema de traducción. Quizá se os había ocurrido desde el principio como la solución más obvia, pero... ¿os habéis parado a pensar en los inconvenientes de la descentralización?

Esto implicaría algún mecanismo para mantener la coherencia de la base de datos de traducciones, ya que ésta ya no se encontraría ubicada en un único lugar, si no repartida a lo largo y ancho del planeta Tierra. De alguna forma, se tiene que conseguir mantener la coherencia en una base de datos en la que aparecen, desaparecen, y se modifican datos constantemente (cada día se conectan nuevas máquinas a la red, y otras tantas cambian de dirección).
La solución empleada consiste en una estructura jerárquica tal y como se muestra en el siguiente gráfico:

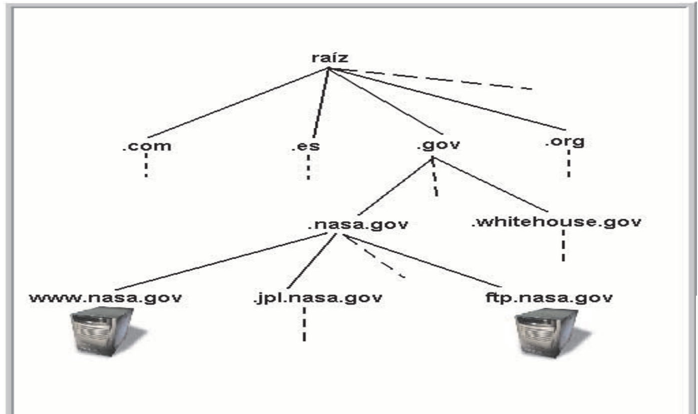


Fig 3.- Ejemplo de jerarquía del sistema DNS.

Como vemos, en primer lugar existe un dominio raíz (root, también representado por un simple punto "."). En este dominio raíz existen actualmente 14 servidores que contienen todos ellos una base de datos con las direcciones IP de los servidores del nivel inmediatamente inferior, es decir, de los servidores de los dominios .com, .net, .org, .es, etc, etc. En estas máquinas no hay más información que esa, por lo que los servidores raíz no tienen ni idea de lo que puede haber por debajo de cada uno de esos dominios. Por ejemplo, ésta podría ser una tabla ficticia de la base de datos de un servidor raíz:

Tabla ficticia de un servidor DNS raíz	
Dominio	***Dirección Servidores de Dominio***
.com	130.25.126.76
	130.25.146.58
	213.121.1.9
.es	215.12.35.42
	221.241.54.2
.org	148.125.42.54
	113.24.123.54
.gov	125.124.32.1

Esas direcciones IP corresponden a servidores DNS de cada uno de esos dominios. Estos dominios que están por debajo del raíz se denominan **TLD** (**T**op **L**evel **D**omain), y existen más de 200 actualmente. Podéis ver una lista de los TLDs correspondientes a cada país en http://www.iso.ch/iso/en/prods-services/iso3166ma/02iso-3166-code-lists/list-en1.html. Aparte de estos, están los TLDs genéricos, como .com, .edu, .gov, etc. Para cada TLD, habrá varios servidores que mantendrán una misma copia de otras tablas, que serán las tablas con las direcciones de los servidores DNS de los niveles inferiores. Por ejemplo, para un servidor .gov habría una entrada correspondiente al servidor de DNS del dominio nasa.gov, y otra para el servidor de DNS del dominio whitehouse.gov:

Tabla ficticia de un servidor de DNS del dominio .gov	
Dominio	***Dirección de Servidores de Dominio***
nasa.gov	125.124.30.1
	98.15.231.2
whitehouse.gov	125.124.31.1
	54.123.12.2
	13.125.42.230

En el dominio nasa.gov, por ejemplo, podríamos encontrarnos con que los servidores son ya una hoja, es decir, un extremo final dentro de la jerarquía en árbol del sistema DNS. Por tanto, aquí las tablas ya no se referirían a otros servidores DNS de subdominios inferiores, si no ya a máquinas concretas. Por ejemplo:

Tabla ficticia de un servidor DNS del dominio nasa.gov	
Dominio	***Dirección de máquinas***
www.nasa.gov	98.15.212.2
	98.15.212.4
jpl.nasa.gov	125.54.23.53
ftp.nasa.gov	98.14.12.1
	125.54.32.2

También podría tratarse de una tabla mixta, en la cual, por ejemplo, la entrada jpl.nasa.gov correspondiese a otro servidor de dominio, y el resto de entradas de la tabla fuesen ya máquinas concretas.

Por tanto, el concepto fundamental que hay que comprender es que cada nivel de la jerarquía **delega** por completo en los niveles inferiores, por lo que es problema de nasa.gov el cómo gestione su nivel y, si quisiera, podría añadir o eliminar nuevos subniveles en cualquier momento, sin tener que informar al nivel raíz, ni a ningún otro nivel superior a él.

Pero entonces, cuando nosotros configuramos nuestra conexión a Internet, y tenemos que especificar las direcciones de los servidores DNS, ¿estamos introduciendo las IPs de los servidores raíz? Este sería un razonamiento

lógico por lo que hemos visto hasta ahora, pero en realidad sería una solución muy poco efectiva, debido principalmente a dos problemas.

En primer lugar, los cientos de millones de usuarios de Internet tendrían que acceder a alguno de los 14 servidores raíz cada vez que quisieran solicitar una traducción. Teniendo en cuenta el sistema de cache, esto sólo ocurriría pocas veces, ya que existen relativamente pocos dominios de primer nivel, o TLDs (.com, .es, .gov, etc), y en cuanto se accediese a una sola dirección de un determinado dominio, ya quedaría almacenada en nuestra cache la dirección de su servidor de dominio.

Pero según fuésemos bajando en la jerarquía irían ocurriendo cada vez más y más solicitudes por cada usuario, y también menos coincidencias de cache.

Por tanto, al final, se tendría una red saturada de peticiones redundantes, en las cuales millones de usuarios pedirían una y otra vez las mismas direcciones (pensad, por ejemplo, cuántos usuarios diferentes habrán solicitado alguna vez la traducción de www.google.com, www.hotmail.com, o cualquier otra dirección popular).

Pero, aparte de la saturación de la red, estaría el problema de la saturación del propio cliente, ya que éste tendría que realizar varias peticiones cada vez que tuviese que realizar una traducción.

Por ejemplo, si tuviese que traducir el nombre antwrp.gsfc.nasa.gov tendría que realizar en primer lugar una conexión con el servidor raíz, en segundo lugar una conexión con el servidor .gov, en tercer lugar, con nasa.gov, y por último con gsfc.nasa.gov, el cual le daría la IP correspondiente a ese nombre. Desde luego, este sistema sería muy poco óptimo.



Fig 4.- Ventana de configuración de servidores DNS al instalar una conexión a Internet con un ISP.

Esas IPs no corresponden a servidores raíz, si no a servidores del propio ISP.

La solución a todo esto consiste en utilizar servidores de DNS comunes a todos los usuarios de un mismo **ISP** (proveedor de Internet, como Telefónica, Ya, Madritel, Ono, etc). Por tanto, cuando un usuario de un ISP quiera realizar una traducción, la solicitará siempre al servidor DNS de su ISP. Éste será el que realice todas las tareas anteriormente descritas. Pero, ¿cual será la gran ventaja? Que al ser este servidor común para miles de usuarios a los cuales habrá atendido previamente sus peticiones, dispondrá el servidor de una cache bastante completa que reducirá enormemente el número de accesos a otros servidores DNS. Todas las páginas populares (como Google, Hotmail, etc), así como las direcciones de los servidores DNS de los TLDs (como los .com, .net, .org...) se encuentran sin duda en la cache de los servidores DNS de todos los ISPs, y su traducción normalmente no tiene que ser solicitada a los servidores de dominio correspondientes.

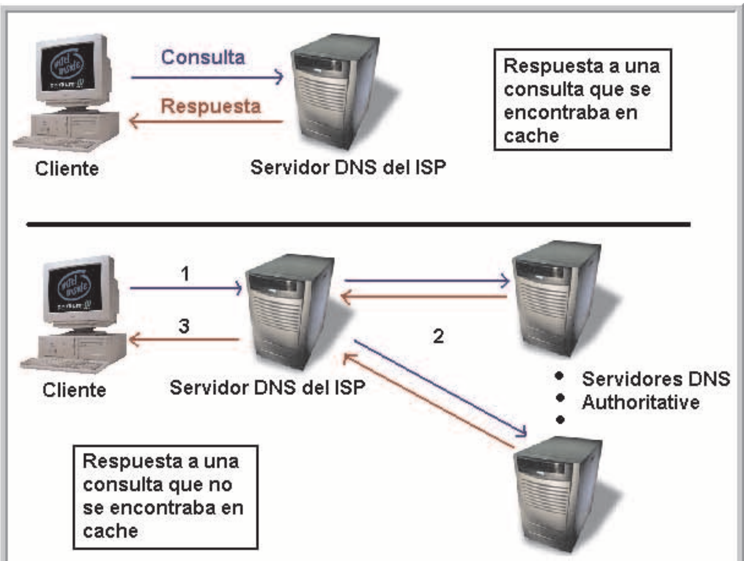


Fig 5.- Interacción típica entre un usuario y el servidor DNS de su ISP

Por supuesto, todo esto representaría una situación ideal, pero en la práctica los contenidos de cache muchas veces no reflejan la realidad, al haberse efectuado cambios, por lo que los servidores DNS han de encargarse de actualizar sus contenidos de cache con los servidores que garantizan que sus contenidos están actualizados, que son los llamados **servidores authoritative** de cada subdominio.

Estos servidores authoritative, que son los que guardan las tablas de traducción reales para cada dominio (y no copias de cache), se dividen en dos tipos: **maestro**, y **esclavos**. Para cada dominio habrá un único servidor authoritative maestro, y uno o varios esclavos. Cada entrada de las tablas de traducción tiene asociado un parámetro **TTL** (**T**ime **T**o **L**ive) que indica durante cuánto tiempo se garantiza que esa información es correcta. Cuando se supere el tiempo de TTL de una entrada, los servidores esclavos conectarán con el maestro para comprobar si los datos que contienen son correctos o han sido modificados. Por tanto, basta con que se realicen todos los cambios en el servidor authoritative maestro para que estos se **propaguen** automáticamente al resto de servidores authoritative del dominio.

3.2. ¿Qué ocurre cuando realizamos una petición de traducción?

Teniendo en cuenta esta estructura jerárquica, vamos a ver el proceso completo desde que solicitamos, por ejemplo, una página web, hasta que conocemos su dirección IP para que nuestro navegador pueda realizar la conexión. Veamos que pasaría si, por ejemplo, pusieramos en nuestro navegador la URL: http://neworder.box.sk/smsread.php?newsid=6586

3.2.1. Nuestro navegador analiza la URL

No todo lo que hay en esa URL es susceptible de ser preguntado al servidor DNS. Para empezar, el prefijo HTTP:// es sólo una información para que el navegador sepa con qué protocolo debe acceder a esos contenidos. En segundo lugar, el sufijo /smsread.php?newsid=6586 es sólo un nombre de directorio, que el navegador debe conocer para solicitarlo al servidor web, una vez que tenga la IP del mismo, tal y como vimos en el artículo sobre HTTP del último número de la revista.
Por tanto, la consulta que tendrá que realizar finalmente el navegador al servidor DNS será esta: neworder.box.sk.

3.2.2. Nuestro navegador solicita la traducción al servidor DNS de nuestro ISP

Aquí terminaría la parte visible del proceso para nosotros, ya que los siguientes pasos serían asunto de otras máquinas que se pelearían para conseguir una respuesta, mientras nosotros nos rascamos la barriga tranquilamente. Por tanto, todos los esfuerzos de nuestro navegador consisten en solicitar al servidor DNS de nuestro ISP la traducción del nombre que indicamos en el anterior paso.
Esto es lo que se llama una traducción **recursiva**, ya que nosotros solicitamos la traducción de un nombre, y ésta nos es devuelta en una sola respuesta, sin que tengamos que seguir nosotros uno a uno todos los pasos para realizar la traducción.

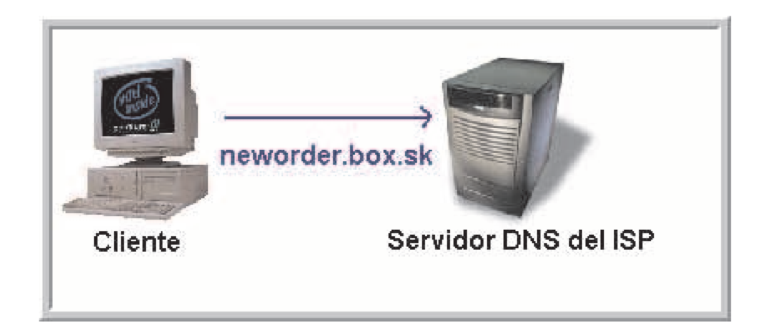


Fig 6.- Nuestra máquina envía un datagrama UDP al servidor DNS de nuestro ISP solicitando la traducción del nombre.

3.2.3. El servidor DNS de nuestro ISP busca en su cache

Ahora la pelota ya está en el tejado del servidor DNS de nuestro ISP, pero como éste es muy listo, antes de armar más jaleo y pasar la pelota a otro, lo primero que hace es buscar en su cache a ver si algún otro cliente (o quizá tú mismo en el pasado) ha solicitado antes esa traducción, por lo que podría darnos una respuesta inmediata.
Si no encuentra ese nombre en su cache, no tendrá más remedio que pasar la pelota.

7.- El servidor nos responde directamente con la traducción, ya que tenía el dato en cache.

Suponiendo que no hubiera encontrado la traducción en su cache, tendría que hacer una serie de consultas bajando por la jerarquía en árbol hasta llegar a la traducción del nombre deseado. Al tener que realizarlo paso a paso, se trata de una traducción **iterativa**, al contrario de la traducción recursiva que solicitábamos nosotros como clientes del ISP.
Lo más lógico sería que el servidor de nuestro ISP empezase por buscar en su cache si tiene la dirección del servidor DNS del dominio .box.sk. En caso de que no fuese así, buscaría la dirección del servidor DNS del dominio .sk, que es ya un TLD.
Suponiendo que no tuviese en cache ninguno de estos datos, continuamos viendo el proceso.

3.2.4. El servidor DNS de nuestro ISP consulta a un servidor del dominio raíz

Nuestro servidor tendrá que empezar por conectarse con uno de los 14 servidores raíz para solicitar las direcciones de los servidores DNS del TLD (dominio de primer nivel) .sk. No tiene problemas para hacer esto, ya que todos los servidores DNS, aunque tengan su cache totalmente vacía, conocen siempre las direcciones de los servidores raíz. Es responsabilidad de los administradores de los servidores DNS el mantener al día las direcciones de los servidores raíz.



Fig 8.- El servidor de nuestro ISP consulta con el servidor raíz, y éste le da la lista de servidores del TLD .sk

El servidor raíz le devolvería una lista con las direcciones de todos los servidores DNS del TLD .sk. De esa lista, nuestro servidor escogería uno cualquiera, y continuaría el proceso. El algoritmo de nuestro servidor para escoger una de las direcciones podría ser tan simple como escoger siempre el primero de la lista que le devuelva el servidor raíz. Esto podría suponer un serio problema, ya que la carga del primer servidor sería enorme, mientras que los demás servidores prácticamente no se usarían. Para evitar esto, el servidor raíz no nos dará la lista siempre en el mismo orden, si no que la

dará en un orden aleatorio, por lo que el primero de la lista será siempre uno diferente. Normalmente, estas direcciones que nos ha devuelto el servidor raíz quedarán almacenadas en la cache de nuestro servidor.

3.2.5. El servidor DNS de nuestro ISP consulta al servidor DNS del TLD .sk

Una vez escogido un servidor DNS del TLD, consulta con éste el siguiente paso en la jerarquía, es decir, la dirección de los servidores DNS del dominio .box.sk.
De nuevo, nuestro servidor escogerá uno de la lista para la siguiente consulta, y almacenará la lista devuelta en su cache.

Fig 9.- Nuestro ISP consulta con el servidor del TLD .sk, el cual le da la lista de servidores del dominio .box.sk.

3.2.6. El servidor DNS de nuestro ISP consulta al servidor DNS de .box.sk

Por último, sólo falta consultar al servidor que definitivamente debería conocer la IP de la máquina a la que queremos acceder: neworder.box.sk. En esta ocasión, el servidor DNS del dominio .box.sk normalmente nos devolverá una sola IP, aunque si se trata de páginas con mucha carga, podría estar repartida ésta en varias máquinas con copias exactas (mirrors) de la página, por lo que nos devolverían una lista de IPs, de la cual escogería una nuestro servidor.

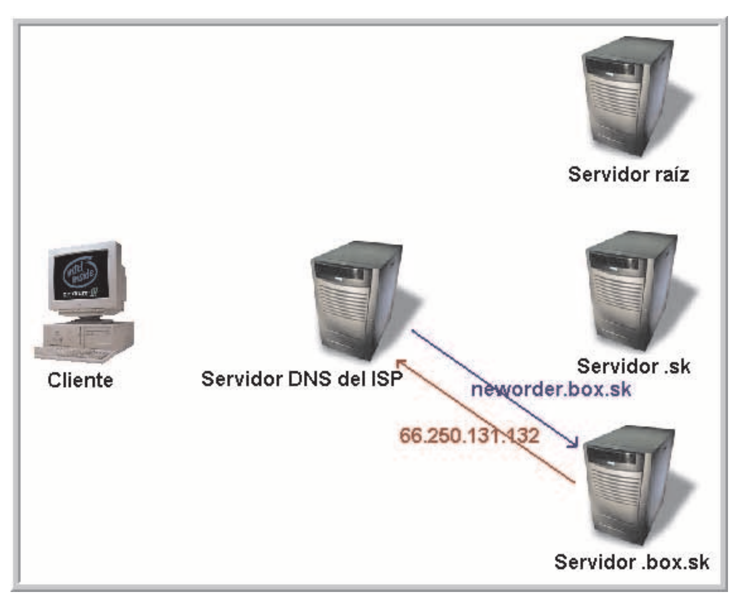


Fig 10.- Nuestro ISP consulta con el servidor de .box.sk, y éste finalmente le da la IP de neworder.box.sk

3.2.7. El servidor DNS de nuestro ISP nos devuelve la traducción

Una vez que ya ha completado todas las consultas iterativas, nuestro servidor DNS nos enviará un mensaje UDP conteniendo la respuesta a nuestra consulta. :-D



Fig 11.- El servidor de nuestro ISP nos responde con un datagrama UDP.

3.3. Tipos de recursos en DNS

Según leíais todo lo anteriormente explicado, probablemente os habrá surgido una duda, y es: ¿cómo sabemos si la respuesta que nos da un servidor es la IP de otro servidor o bien la IP de una máquina? Eso quizá sea posible

saberlo según el contexto, pero los datos que están almacenados en las tablas de los servidores DNS carecen de ningún contexto, por lo que tiene que estar también almacenado con cada entrada de la tabla un dato que indique a qué se está refiriendo.

El tipo de cada entrada de estas tablas es uno de los parámetros de lo que se define como **RR** (**R**esource **R**ecord), y es un dato que ha de ser enviado en cada respuesta junto con la traducción.

Pero no sólo existen dos tipos de RRs que definan si se trata de una IP de una máquina o de un servidor DNS, si no muchos otros.

La lista completa la tenéis en http://www.iana.org/assignments/dns-parameters , pero tranquilos, que os voy a resumir aquí los más importantes. :-)

NS (Name Server) – La dirección se refiere a la IP de un servidor DNS. (ej: .box.sk)

A (Address) – La dirección se refiere a la IP de una máquina. (ej: neworder.box.sk)

PTR (Pointer) – Se trata de una entrada inversa, es decir, en lugar de ser la IP correspondiente a un nombre, es el nombre correspondiente a una IP.

MX (Mail Exchange) – La dirección se refiere al servidor de correo de un dominio concreto. vimos algo sobre esto en el artículo sobre SMTP, en el número 8 de la revista.

Según el tipo de RR, la cabecera será diferente. Se pueden ver las cabeceras para todos los tipos en el **RFC 1035**.

Por tanto, en nuestro ejemplo anterior, el usuario consultaba al servidor DNS de su ISP una dirección de **tipo A**, mientras que el servidor DNS de su ISP realizaba una serie de consultas de **tipo NS** a varios servidores, hasta que al final al último le hacía una consulta de una dirección **tipo A**.

3.4. Detalles del protocolo RAW

Para ver en "crudo" el protocolo DNS vamos a utilizar ingeniería inversa para analizar una "sesión" DNS. Pongo sesión entre comillas porque debemos recordar que DNS funciona mediante UDP, por lo que no existe una conexión establecida, si no que cada mensaje es independiente. Yo he usado para el ejemplo el sniffer **Ethereal**, que es un sniffer gráfico para Linux. Simplemente he puesto en captura el sniffer y he escrito en la consola de Linux:

host www.hackxcrack.com

Este comando sirve típicamente para resolver un nombre. Suponiendo que la dirección solicitada no se encuentre en el archivo /etc/hosts ni en ningún tipo de cache, este comando lanzará una solicitud de traducción a un servidor DNS, el que esté configurado por defecto en el sistema.

3.4.1. Consulta

Vamos a ver la captura de pantalla de Ethereal:

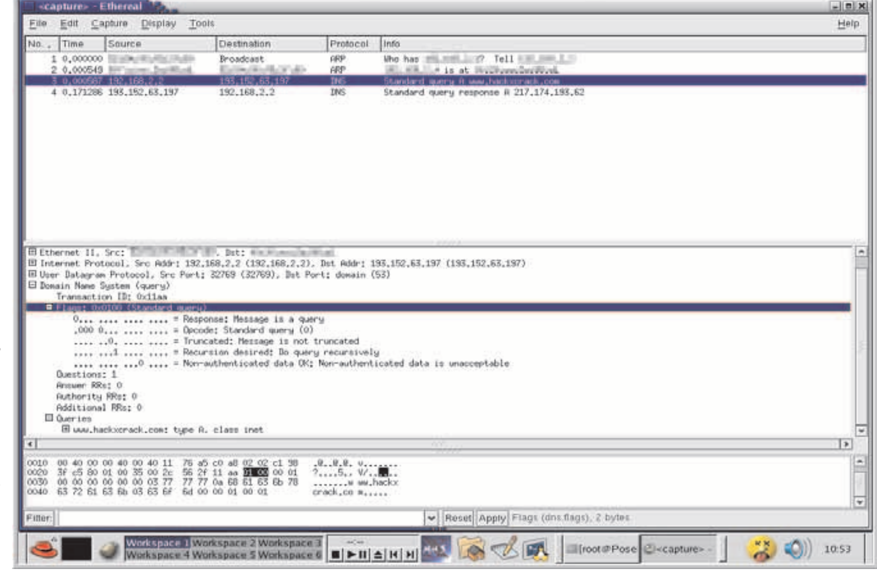


Fig 12.- Captura de pantalla de Ethereal, mostrando la consulta realizada al servidor DNS.

En la ventana de arriba tenemos la lista de paquetes recibidos. Los dos primeros no tienen ningún interes, ya que es simplemente una consulta de ARP a la dirección broadcast para buscar la dirección MAC de mi router. Si no habéis entendido ni una palabra, quizá es que haría falta algún artículo sobre el protocolo MAC. ;-)

Los que nos interesan ahora son los dos paquetes siguientes. Si hemos visto capturas de sesiones TCP sabremos que suele haber una serie de paquetes adicionales que no contienen datos, que son los que se utilizan para establecer y para cerrar la conexión. En este caso vemos que la cosa es mucho más sencilla, ya que sólo hay 2 paquetes: una consulta, y su consiguiente respuesta.

Esto, por supuesto, es debido a que se trata del protocolo **UDP**, que no es orientado a conexión.

En la ventana de en medio vemos los detalles del paquete UDP que hemos enviado nosotros como consulta.

Como vemos, no sólo se envía el nombre que queremos traducir, si no también una cabecera con una serie de datos necesarios para la consulta.

El primer dato de la cabecera es el **identificador de transacción** (**transaction ID**) que consiste en 2 bytes que identifican un par consulta-respuesta, es decir, la respuesta a esta consulta tendrá que tener el mismo identificador de transacción, para que el cliente sepa que esa respuesta es precisamente la que está esperando a esa pregunta.

Vemos que se envían también dos bytes con una serie de **flags**, es decir, una serie de bits que, según estén a 0 o a 1 tendrán un significado u otro.

El primer flag indica que el paquete enviado se trata de una **consulta**, y no de una respuesta. Para ello, hay que poner ese flag a **0**.

El segundo flag se codifica mediante 4 bits. En este caso, **0000**, se trata de una **consulta estándar**, que son las que utilizaremos normalmente.

Después, tenemos un flag a **0** que indica que el mensaje **no está truncado**, lo cual es lo habitual.

Después, vemos un flag con valor **1** que indica que la solicitud que hacemos queremos que sea **recursiva**. Si estuviese a 0, la solicitud sería iterativa, y en ese caso nuestra máquina tendría que realizar uno a uno todos los pasos para resolver el nombre, tal y como vimos que hacía el servidor DNS de nuestro ISP. Como, por supuesto, queremos delegar esta ardua tarea en nuestro ISP, que para eso está, activaremos este flag.

Finalmente, el último flag indica que exigimos que la respuesta contenga información para garantizar su autenticidad.

Después de los flags hay 4 campos más que indican cuál será el contenido de los datos siguientes. En este caso, se tratará únicamente de una consulta.

Después de la cabecera, está el **nombre** de la consulta: www.hackxcrack.com, y por último se codifica el tipo de consulta que, en este caso, es **tipo A**.

3.4.2. Respuesta

Con respecto a la respuesta del servidor, hay que explicar unas cuantas cosas más.

```
⊟ Domain Name System (response)
    Transaction ID: 0x11aa
  ⊟ Flags: 0x8180 (Standard query response, No error)
      1... .... .... .... = Response: Message is a response
      .000 0... .... .... = Opcode: Standard query (0)
      .... .0.. .... .... = Authoritative: Server is not an authority for domain
      .... ..0. .... .... = Truncated: Message is not truncated
      .... ...1 .... .... = Recursion desired: Do query recursively
      .... .... 1... .... = Recursion available: Server can do recursive queries
      .... .... ..0. .... = Answer authenticated: Answer/authority portion was not authenticated by the server
      .... .... .... 0000 = Reply code: No error (0)
    Questions: 1
    Answer RRs: 1
    Authority RRs: 2
    Additional RRs: 2
  ⊟ Queries
    ⊟ www.hackxcrack.com: type A, class inet
```

Fig 13.- Cabecera de la respuesta.

En primer lugar, podemos comprobar que, en efecto, el **identificador de transacción** es el mismo que el de la consulta anterior (**11aa**, en hexadecimal).

A continuación, tenemos los dos bytes de **flags**, cuyo significado en este caso es diferente. Al ser **1** el primer bit, lo cual quiere decir que se trata de una **respuesta**, y no de una consulta, el significado del resto de bits se interpretará de forma diferente.

Después de los 4 bits del identificador del **tipo de consulta** vemos que hay un bit a **0** que indica que el servidor DNS **no es authoritative**, lo cual es lógico, ya que se trata del servidor DNS de nuestro ISP, y éste normalmente no será authoritative para los dominios que consultemos. El próximo flag nuevo que vemos es el **llamado Recursion Available** que, como vemos, está a **1** ya que, por supuesto, el servidor DNS de nuestro ISP tiene que **aceptar solicitudes recursivas**, que son las que le harán siempre sus clientes. El siguiente flag, **Answer authenticated**, indica que no se ha utilizado **autenticación**, lo cual es un asunto que no veremos por falta de espacio. Por último, un **código de error** codificado con **0000** nos indica que todo ha ido correctamente y no hay **ningún error**.

Los próximos 4 campos nos indican que aparte de esta cabecera habrá un RR de respuesta, dos RRs de autoridad, y dos RRs adicionales. Por supuesto, vamos a ver todo esto ahora mismo.

El campo de datos comienza con una replicación de la consulta que realizamos nosotros previamente.

A continuación, se encuentra el primer **RR**, que es el de **respuesta**.

```
⊟ Answers
   ⊟ www.hackxcrack.com: type A, class inet, addr 217.174.193.62
        Name: www.hackxcrack.com
        Type: Host address
        Class: inet
        Time to live: 1 hour, 55 minutes, 11 seconds
        Data length: 4
        Addr: 217.174.193.62
⊟ Authoritative nameservers
   ⊞ hackxcrack.com: type NS, class inet, ns ns1.amenworld.com
   ⊞ hackxcrack.com: type NS, class inet, ns ns2.amenworld.com
⊟ Additional records
   ⊞ ns1.amenworld.com: type A, class inet, addr 217.174.192.237
   ⊞ ns2.amenworld.com: type A, class inet, addr 195.154.205.8
```

Fig 14.- RR de respuesta.

Como vemos, el **tipo** de este RR es **A (Host Address)**, y vemos que se encuentra también el campo **TTL** (**T**ime **T**o **L**ive) del que hablamos anteriormente, que indica durante cuánto tiempo, como máximo, será valida esta traducción. La IP 217.174.193.62 es la tan ansiada respuesta a nuestra consulta, es decir, la IP asociada al nombre www.hackxcrack.com.

Después del RR de respuesta, se encuentran dos **RRs de autoridad**, que son los que exigimos en nuestra consulta para garantizar la autenticidad de los datos.

```
⊟ Answers
   ⊞ www.hackxcrack.com: type A, class inet, addr 217.174.193.62
⊟ Authoritative nameservers
   ⊟ hackxcrack.com: type NS, class inet, ns ns1.amenworld.com
        Name: hackxcrack.com
        Type: Authoritative name server
        Class: inet
        Time to live: 1 day, 1 hour, 55 minutes, 11 seconds
        Data length: 16
        Name server: ns1.amenworld.com
   ⊟ hackxcrack.com: type NS, class inet, ns ns2.amenworld.com
        Name: hackxcrack.com
        Type: Authoritative name server
        Class: inet
        Time to live: 1 day, 1 hour, 55 minutes, 11 seconds
        Data length: 6
        Name server: ns2.amenworld.com
```

Fig 15.- RRs de autoridad.

Como vemos, el **tipo** de estos RRs es **NS**, ya que se trata de servidores DNS, y no de direcciones de máquinas. En estos RRs de autoridad sólo se incluye los nombres de los servidores authoritative, pero no sus direcciones IP. Éstas se encuentran traducidas en los 2 **RRs adicionales** que completan la respuesta.

```
⊟ Additional records
    ⊟ ns1.amenworld.com: type A, class inet, addr 217.174.192.237
          Name: ns1.amenworld.com
          Type: Host address
          Class: inet
          Time to live: 22 hours, 14 minutes, 48 seconds
          Data length: 4
          Addr: 217.174.192.237
    ⊟ ns2.amenworld.com: type A, class inet, addr 195.154.205.8
          Name: ns2.amenworld.com
          Type: Host address
          Class: inet
          Time to live: 1 hour, 39 minutes, 30 seconds
          Data length: 4
          Addr: 195.154.205.8
```

Fig 16.- RRs adicionales.

Con todo lo que hemos visto hasta ahora, nos damos cuenta de que el protocolo DNS es complicado de manejar en su forma cruda (RAW), ya que la mayoría de los campos se encuentran codificados, así que he considerado que, más interesante que saber manejarlo, es el saber interpretarlo a grandes rasgos.

Supongo que recordaréis que he mencionado que en ciertas ocasiones se utilizaba **TCP** en lugar de UDP. Esto suele ocurrir en 2 situaciones concretas. En primer lugar, en las transacciones que realizan los servidores de un dominio para mantener la coherencia en sus bases de datos (esclavos solicitando entradas caducadas al maestro), y en segundo lugar en las solicitudes con una longitud mayor de 512 bytes, lo cual es poco habitual.

4. Algunos problemas de seguridad del sistema DNS

4.1. Ataques DoS

El 21 de Octubre del año 2002 se registró uno de los mayores ataques contra toda la red Internet, que consistió precisamente en un ataque DoS distribuido contra los 13 servidores raíz del sistema DNS. El ataque duró aproximadamente una hora, y "sólo" consiguió afectar con cierta seriedad a 7 servidores.

Debido a esto, las consecuencias del ataque no fueron demasiado importantes, ya que sólo un pequeño porcentaje de solicitudes quedó sin respuesta durante ese breve periodo de una hora.

En principio, nos puede parecer que debió tratarse de un problema terrible, pero en realidad no lo fue tanto. En primer lugar, tal y como ya hemos visto, el número de accesos directos a los servidores raíz es relativamente pequeño, ya que la mayoría de los servidores DNS ya han realizado gran número de consultas previas a los servidores raíz, por lo que tienen en su cache prácticamente cualquier consulta que pudiesen necesitar de los servidores raíz. Por supuesto, estas caches tienen un tiempo de vida (recordemos el parámetro **TTL**), pero en sólo una hora que duró el ataque el número de nuevas consultas tuvo que ser muy reducido. Por otra parte, los expertos afirman que el sistema DNS puede funcionar sin problemas siempre y cuando funcionen **5** de los 14 servidores raíz, y en este caso no se vieron afectados 6 servidores, por lo que el sistema podía soportar la carga.

Un problema potencial de los servidores raíz es que no tienen una **distribución geográfica** adecuada, ya que 10 de ellos están situados en Estados Unidos. Actualmente, está en marcha un plan para mejorar esta distribución y, por ejemplo, el decimo cuarto servidor raíz fue instalado nada menos que en España. :-D

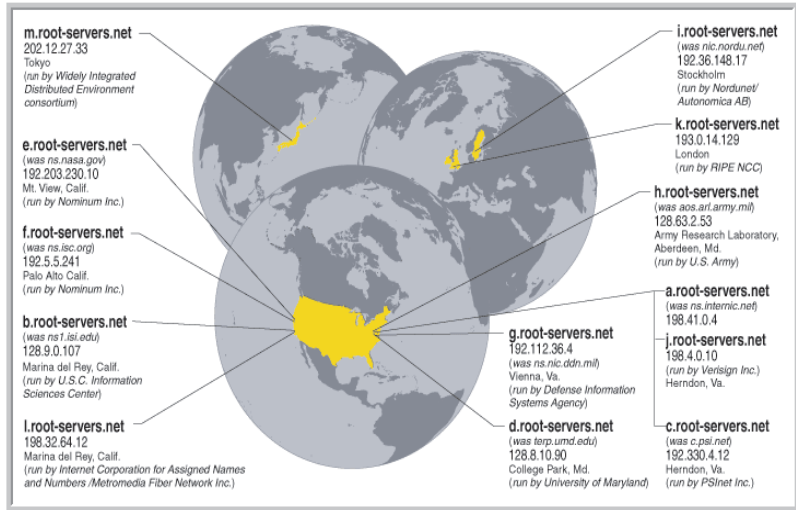


Fig 17.- Localización geográfica de 13 de los 14 servidores raíz del sistema DNS. El decimo cuarto se ubica en España.

4.2. Envenenamiento de cache DNS

Llegamos al fin a una técnica realmente interesante y bastante avanzada, pero muy peligrosa por sus consecuencias, y que no recomiendo en absoluto que sea utilizada.

¿Se os ha ocurrido pensar que ocurriría si alguien consiguiese engañar al servidor DNS de vuestro ISP al hacerse pasar por un servidor authoritative? Pues, como habréis adivinado, la cache de vuestro servidor se llenaría de traducciones falsas que él consideraría como válidas.

Cuando vosotros o cualquier otro cliente solicitase una de estas traducciones a vuestro servidor, éste os respondería con la traducción falsa que almacenó en su cache. Imaginad por ejemplo que la IP que solicitáis es la de vuestro banco, o la de vuestra cuenta de correo.

El atacante podría haber creado una página exactamente con la misma apariencia que la página de LOGIN para entrar en vuestra cuenta del banco o vuestra cuenta de correo, y haber ubicado esa página en su propia máquina. Bastaría con que envenenase la cache de vuestro servidor DNS haciéndole creer que ese banco o ese servidor de correo se ubicaba en su propia máquina, para que cualquier usuario que intentase acceder a esa página accediese en realidad a la máquina del atacante, introduciendo confiadamente sus datos de usuario y password.

Por supuesto, como pillen al atacante se le puede caer el pelo, ya que puede causar auténticos estragos, teniendo en cuenta que un servidor DNS puede servir a miles de usuarios.

Por otra parte, otro inconveniente para el atacante es que tiene que repetir el mismo ataque cada cierto tiempo, ya que los contenidos en cache, como ya sabemos, tienen un tiempo de vida (TTL), y transcurrido ese tiempo el servidor envenenado volvería a solicitar las traducciones caducadas.



Fig 18.- Esquema básico de un envenenamiento de cache. Si la respuesta del atacante llega antes que la del servidor authoritative, será ésta la que se tome por válida.

Pero, ¿cómo es posible engañar a un servidor DNS? Pues la respuesta depende de qué software utilice el servidor. La inmensa mayoría de servidores DNS del mundo utilizan el popular software **BIND**, por lo que nos centraremos sólo en éste.

Para responder a esta pregunta tendremos que empezar por analizar el problema. Pensemos antes de nada en los mecanismos que se utilizan para comprobar la autenticidad de una respuesta DNS. En primer lugar, al utilizarse como protocolo de transporte **UDP**, no habrá que "colarse" dentro de ninguna conexión, lo cual es la primera ayuda para el atacante. Colarse en una sesión **TCP** es bastante complicado, ya que no basta con conocer las **IPs** de origen y destino, así como los **puertos** de origen y destino, si no que también es preciso conocer los **números de secuencia** TCP. Sería necesario un artículo sobre el protocolo TCP para que comprendáis esto, así que quedaos simplemente con la idea de que es muy complicado colarse dentro de una conexión TCP. En cambio, en el caso de **UDP**, no hay números de secuencia, por lo que basta con saber las **IPs** de origen y destino, así como los **puertos** de origen y destino.

En el caso que nos ocupa, la **IP de destino** será la del **servidor víctima**, ya que éste será el que realizará una consulta al servidor authoritative, cuya respuesta dará el atacante en lugar de éste.

La **IP de origen** es la del **servidor authoritative** auténtico, que el atacante tendrá que **spoofear**, es decir, enviar sus paquetes con esa IP en lugar de con su propia IP.

Se sale del tema del artículo explicar las técnicas de **IP spoofing**.

Con respecto a los **puertos**, el tema es bastante más complicado, ya que si bien el puerto de **origen** sabemos que es el **53** (puerto UDP de DNS) el puerto de **destino** será un puerto "**aleatorio**" que utilizó el servidor víctima para su consulta.

En la práctica existen formas de averiguar este puerto, debido a que BIND no utiliza siempre puertos aleatorios para sus consultas, si no que repite el mismo número de puerto.

¿Y ya está todo con eso? Bueno, hemos hablado de como spoofear una respuesta UDP genérica, pero en nuestro caso nos encontramos con el caso concreto del protocolo **DNS**, y este protocolo añade un elemento adicional de "seguridad", que es el de los **identificadores de transacción** (**transaction IDs**).

Una respuesta, aunque teóricamente provenga del servidor authoritative y utilice el mismo puerto UDP, no será aceptada como válida si tiene un identificador de transacción diferente al de la consulta. Y aquí, en la predicción del identificador de transacción, es donde el tema depende de qué software utilice el servidor víctima.

Hasta el año 1997, cuando el **CERT** (**C**omputer **E**mergency **R**esponse **T**eam) comunicó el pertinente aviso de seguridad sobre BIND, éste utilizaba identificadores de transacción **secuenciales**, y no aleatorios, por lo que predecir el identificador era una tarea totalmente trivial.

En cambio, actualmente es de esperar que todos los servidores hayan actualizado sus versiones de BIND y no creo que quede en el mundo uno solo que aún utilice identificadores secuenciales.

A pesar de que los identificadores pasaron a ser aleatorios, existen gran cantidad de teorías matemáticas y estadísticas sobre los números **pseudo-aleatorios**, que son los que en realidad genera un ordenador, y estas teorías permiten calcular la probabilidad de éxito según el sistema de generación de números utilizado.

En primer lugar, aunque no hay que contar con ello si el servidor cuenta con un administrador eficiente, BIND puede generar **múltiples consultas** para consultar un sólo dominio, por lo que recibirá varias respuestas del mismo servidor authoritative.

Esto da lugar a que el atacante no tenga que adivinar un identificador de transacción único, si no que le bastaría con adivinar uno de los muchos identificadores que habría utilizado el servidor en sus múltiples consultas.

Analizando esto desde el punto de vista estadístico, nos encontramos con la interesante "**paradoja del cumpleaños**" (birthday paradox) que afirma que en un grupo de 23 personas, la probabilidad de que dos de ellas tengan la misma fecha de cumpleaños es superior al 50%.

A mi me gusta más expresar esta idea desde un punto de vista mucho más gráfico, que consiste en imaginar un parking con 365 plazas, en el cual entrasen 23 coches con las luces apagadas, y pensar en la probabilidad de que dos de ellos se estrellasen al intentar ocupar la misma plaza.

Esta misma idea estadística se puede aplicar al caso que nos ocupa aunque, por supuesto, manejando diferentes cifras.

Así, se llega a la conclusión de que con "sólo" enviar **700** respuestas spoofeadas, la probabilidad de que alguna de ellas coincida en el identificador de transacción es cercana al **100%.**

Fig 19.- Esquema detallado de un ataque de envenenamiento. En el paso 1, el atacante hace repetidas solicitudes del dominio que se quiere envenenar. En el paso 2, el servidor de la víctima consulta ese dominio repetidas veces con el servidor authoritative. En el paso 3, es el atacante el que responde haciéndose pasar por el servidor authoritative. En este caso, el atacante se ha ayudado de un DoS al servidor authoritative que le impide enviar las respuestas legítimas, por lo que no hay conflicto entre respuestas falsas y auténticas.

Pero claro... el servidor authoritative legitimo, y no sólo el atacante, responderá también a la víctima. Aquí nos encontramos con un conflicto en el que unas veces podrá ganar el atacante, y otras el servidor authoritative, según quién consiga dar antes su respuesta. Por eso, este tipo de ataques aumentan mucho su efectividad si se combinan con ataques **DoS** a los servidores authoritative, ralentizando o incluso anulando sus respuestas.

Si tratamos con servidores bien configurados que no hagan múltiples consultas para un sólo dominio, tendremos que entrar de lleno en complicadísimas teorías matemáticas cuya explicación se sale por completo de lo que puede abarcar este artículo. A modo de ejemplo, os comento que con el sistema de generación de números pseudo-aleatorios de **BIND 8.x** se puede tener una certeza del **100%** de adivinar el identificador de transacción con tan sólo **3 paquetes!** :-O

En cambio, con **BIND 9.x** la cosa es ya mucho más complicada, ya que incluso con **5000 paquetes** la probabilidad de éxito es sólo del **20%**. Esto es debido a que el sistema generador de números pseudo-aleatorios es matemáticamente mucho más impredecible y, por tanto, mucho más eficiente.

Autor: PyC (LCo)

SERVIDOR DE HXC MODO DE EMPLEO

- Hack x Crack ha habilitado tres servidores para que puedas realizar las prácticas de hacking.

- Las IPs de los servidores de hacking las encontrarás en EL FORO de la revista (www.hackxcrack.com). Una vez en el foro entra en la zona COMUNICADOS DE HACK X CRACK (arriba del todo) y verás varios comunicados relacionados con los servidores. No ponemos las IP aquí porque es bueno acostumbrarte a entrar en el foro y leer los comunicados. Si hay alguna incidencia o cambio de IP o lo que sea, se comunicará en EL FORO.

- Actualmente tienen el BUG del Code / Decode. La forma de "explotar" este bug la explicamos extensamente en los números 2 y 3. Lo dejaremos así por un tiempo (bastante tiempo ;) Nuestra intención es ir habilitando servidores a medida que os enseñemos distintos tipos de Hack.

- En los Servidores corre el Windows 2000 con el IIS de Servidor Web. No hemos parcheado ningún bug, ni tan siquiera el RPC y por supuesto tampoco hemos instalado ningún Service Pack. Para quien piense que eso es un error (lógico si tenemos en cuenta que el RPC provoca una caída completa del sistema), solo decirte que AZIMUT ha configurado un firewall desde cero que evita el bug del RPC, (bloqueo de los puertos 135 (tcp/udp), 137 (udp), 138 (udp), 445 (tcp), 593 (tcp)). La intención de todo esto es, precisamente, que puedas practicar tanto con el CODE/DECODE como con cualquier otro "bug" que conozcas (y hay cientos!!!). Poco a poco iremos cambiando la configuración en función de la experiencia, la idea es tener los Servidores lo menos parcheados posibles pero mantenerlos operativos las 24 horas del día. Por todo ello y debido a posibles cambios de configuración, no olvides visitar el foro (Zona Comunicados) antes de "penetrar" en nuestros servidores.

- Cada Servidor tiene dos unidades (discos duros duros):
* La unidad c: --> Con 40GB y Raíz del Sistema
* La unidad d: --> Con 40GB
* La unidad e: --> CD-ROM

Nota: Raíz del Servidor, significa que el Windows Advanced Server está instalado en esa unidad (la unidad c:) y concretamente en el directorio por defecto \winnt\ Por lo tanto, la raíz del sistema está en c:\winnt\

- El IIS, Internet Information Server, es el Servidor de páginas Web y tiene su raíz en c:\inetpub (el directorio por defecto)

Nota: Para quien nunca ha tenido instalado el IIS, le será extraño tanto el nombre de esta carpeta (c:\inetpub) como su contenido. Pero bueno, un día de estos os enseñaremos a instalar vuestro propio Servidor Web (IIS) y detallaremos su funcionamiento.

De momento, lo único que hay que saber es que cuando TÚ pongas nuestra IP (la IP de uno de nuestros servidores) en tu navegador (el Internet explorer por ejemplo), lo que estás haciendo realmente es ir al directorio c:\Inetpub\wwwroot\ y leer un archivo llamado default.htm.

Nota: Como curiosidad, te diremos que APACHE es otro Servidor de páginas Web (seguro que has oído hablar de él). Si tuviésemos instalado el apache, cuando pusieses nuestra IP en TU navegador, accederías a un directorio raíz del Apache (donde se hubiese instalado) e intentarías leer una página llamada index.html ... pero... ¿qué te estoy contando?... si has seguido nuestra revista ya dominas de sobras el APACHE ;)

Explicamos esto porque la mayoría, seguro que piensa en un Servidor Web como en algo extraño que no saben ni donde está ni como se accede. Bueno, pues ya sabes dónde se encuentran la mayoría de IIS (en \Inetpub\) y cuál es la página por defecto (\Inetpub\wwwroot\default.htm). Y ahora, piensa un poco… … ¿Cuál es uno de los objetivos de un hacker que quiere decirle al mundo que ha hackeado una Web? Pues está claro, el objetivo es cambiar (o sustituir) el archivo default.html por uno propio donde diga "hola, soy DIOS y he hackeado esta Web" (eso si es un lamer ;)

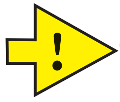
A partir de ese momento, cualquiera que acceda a ese servidor, verá el default.htm modificado para vergüenza del "site" hackeado. Esto es muy genérico pero os dará una idea de cómo funciona esto de hackear Webs ;)

- Cuando accedas a nuestro servidor mediante el CODE / DECODE BUG, crea un directorio con tu nombre (el que mas te guste, no nos des tu DNI) en la unidad d: a ser posible y a partir de ahora utiliza ese directorio para hacer tus prácticas. Ya sabes, subirnos programitas y practicar con ellos :) ... ... ¿cómo? ¿que no sabes crear directorios mediante el CODE/DECODE BUG... repasa los números 2 y tres de Hack x Crack ;p

Puedes crearte tu directorio donde quieras, no es necesario que sea en d:\mellamojuan. Tienes total libertad!!! Una idea es crearlo, por ejemplo, en d:\xxx\system32\default\10019901\mellamojuan (ya irás aprendiendo que cuanto mas oculto mejor :)

Es posiblemente la primera vez que tienes la oportunidad de investigar en un servidor como este sin cometer un delito (nosotros te dejamos y por lo tanto nadie te perseguirá). Aprovecha la oportunidad!!! e investiga mientras dure esta iniciativa (esperemos que muchos años).

- En este momento tenemos mas de 600 carpetas de peña que, como tu, está practicando. Así que haznos caso y crea tu propia carpeta donde trabajar.



MUY IMPORTANTE...

MUY IMPORTANTE!!!!! Por favor, no borres archivos del Servidor si no sabes exactamente lo que estás haciendo ni borres las carpetas de los demás usuarios. Si haces eso, lo único que consigues es que tengamos que reparar el sistema servidor y, mientras tanto, ni tu ni nadie puede disfrutar de él :(Es una tontería intentar "romper" el Servidor, lo hemos puesto para que disfrute todo el mundo sin correr riesgos, para que todo el mundo pueda crearse su carpeta y practicar nuestros ejercicios. En el Servidor no hay ni Warez, ni Programas, ni claves, ni nada de nada que "robar", es un servidor limpio para TI, por lo tanto cuídalo un poquito y montaremos muchos más :)

CURSO DE VISUAL BASIC
UN CLIENTE, UNA NECESIDAD, TENEMOS UN PROYECTO (PARTE II)

Como no, seguimos con nuestro particular curso de Visual Basic.
Esta vez necesitaremos algunos conceptos básicos de acceso a Bases de Datos (SQL). "SELECT", "FROM", "WHERE" ... ...

Os doy la bienvenida a esta segunda parte de mi última entrega. Bien, donde lo dejamos... ¡Ah! , Sí, habíamos acabado el control de Stock. Pero faltaba algo, creo que en el último número dije que había un gazapo en el código de alta. Por supuesto, lo vamos a solucionar. Aquí tenemos el código de alta:

```
Case "NUEVO"
 If CamposOK = True Then
    Rs.MoveFirst
    Rs.AddNew
    MoverDatosARegistro
    Rs.Update
    MsgBox "Registro dado de alta"
    Lista.ListItems.Clear
 End If
 Call Form_Load
```

No es lógico que demos de alta un nuevo producto en el stock si este ya existe, es decir, si tenemos 30 Calipos de menta, e introducimos en la tabla de Stock 10 Calipos de menta, deberíamos tener un total de 40 Calipos de menta y no un registro con 30 y otro con 10 independientes.

Os reté a que lo arreglarais sin mi ayuda, y estoy seguro de que algunos lo conseguisteis. Aquí os voy a contar una manera sencilla aunque no sea la más óptima. En este punto, deberíamos hacer una introducción al SQL. No os voy a dar la lata con historia, pero si deciros que el SQL es un stándard que nos permite realizar consultas sobre una base de datos.

Está compuesto por varias partes:

- La primera es la cláusula "SELECT", donde indicamos los campos que queremos traernos de la tabla, siendo el carácter "*" todos los campos.

- Posteriormente incluimos la cláusula "FROM", donde indicaremos de qué tabla son los campos descritos anteriormente.

- La cláusula "WHERE" nos indica una condición, por ejemplo, "cuando el sabor sea menta".

Estas suelen ser las básicas, aunque la parte que comprende el "WHERE" se puede omitir en caso de que se quiera recoger todos los registros de una tabla. También se pueden hacer ordenaciones o agrupaciones de los registros que nos hemos traído con nuestro SQL.

Voy a poner varios ejemplos para entenderlo mejor. Digamos que al abrir el Recordset, en vez de poner el nombre de la tabla escribimos:

```
Select * from Stock
```

Estaríamos realizando exactamente el mismo efecto que si solo escribiéramos la tabla, pero si añadimos una pequeña cláusula más:

```
Select * from Stock where Nombre='Calipo'
```

Estaríamos recibiendo solo aquellos registros

que tengan en el campo Nombre el literal "Calipo". Analicemos la sentencia:

- "Select *", con esto estamos diciendo que queremos traernos todos los campos de la tabla, en este caso, "Nombre, Sabor , Cantidad, Id".

- La siguiente cláusula: "from Stock". Quiere decir que estamos interactuando con la tabla Stock, y que los datos que vamos a traernos son de esta tabla. "Where Nombre='Calipo'". Esto filtra los datos que nos va a devolver la consulta. Le está diciendo al Recordset que solo traiga aquellos registros con nombre "Calipo".

Para ver como funciona, haremos una prueba temporal con el código que tenemos. Vamos al evento Form_Load() del formulario Stock y cambiamos el código al que vemos a continuación:

```
Private Sub Form_Load()
   Set Rs = New ADODB.Recordset
   Rs.Open "select * from stock where Nombre='Calipo'",
   Conn.ConnectionString, adOpenDynamic, adLockOptimistic
   While Not Rs.EOF
      With Lista.ListItems.Add(, , Rs("Id"))
         .SubItems(1) = Rs("Nombre")
         .SubItems(2) = Rs("Sabor")
         .SubItems(3) = Rs("Cantidad")
      End With
      Rs.MoveNext
   Wend
   MoverDatosACampos
   Deshabilitar
End Sub
```



Recuerda cambiar...

Recuerda cambiar de nuevo el código después de la prueba. Si queréis dejar una consulta SQL en vez del nombre de la tabla, podéis poner "Select * from stock"

Vemos que la única variación está en el "Rs.Open", que hemos cambiado el literal con el nombre de la tabla por la sentencia SQL. Ejecutamos, y si todo ha ido bien, solo deberíamos ver por pantalla aquellos helados que sean "Calipo".

Espero que halláis entendido, muy por encima, el SQL. Podríamos dedicar varios números a esto, pero no creo que sea necesario para este ejercicio.



Puedo decir que...

Puedo decir que en Internet hay auténticos manuales de consulta sobre SQL muy buenos, por si queréis profundizar más en el tema.

Vale, pues entonces, vamos a arreglar esa pequeña errata con una sencilla consulta SQL. Tenemos que codificar el código donde estamos modificando registros de tal manera que, desde ahora, cada vez que se intente dar de alta un helado, y este se encuentre en la tabla, se sumen las cantidades, en lugar de dar de alta uno nuevo. Mi propuesta de código es la siguiente:

```
Case "NUEVO"
   If CamposOK = True Then
      Rs.Close
      Rs.Open "Select * from Stock where Nombre='" &
      Txtnombre & "' and sabor='" & TxtSabor & "'",
      Conn.ConnectionString, adOpenDynamic, adLockOptimistic
      If Not Rs.EOF Then
         Rs("Cantidad") = Rs("Cantidad") + TxtCantidad
         Rs.Update
         MsgBox "Ya se ha encontrado un helado con esas
         caracteristicas, se van a sumar las cantidades"
         Lista.ListItems.Clear
      Else
         Rs.AddNew
         MoverDatosARegistro
         Rs.Update
         MsgBox "Registro dado de alta"
         Lista.ListItems.Clear
      End If
   End If
   Call Form_Load
```

Pasamos a comentarlo:

La comprobación de campos la dejamos tal cual, pero inmediatamente después cerramos el Recordset para abrirlo posteriormente con una SQL que busque un helado con los mismos valores que los que tenemos en las cajas de texto.

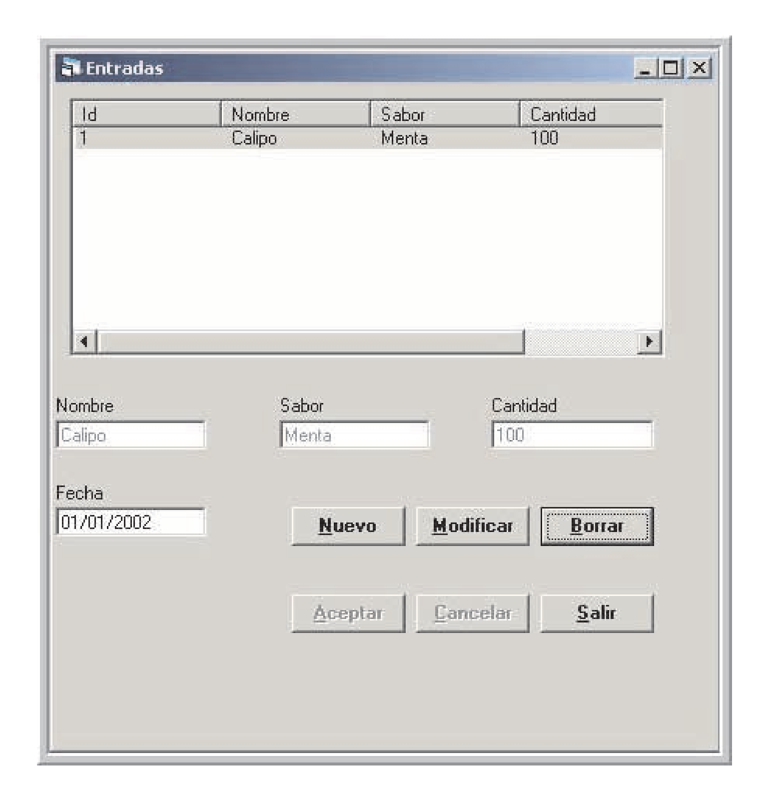
Podemos observar que en la cláusula "WHERE" de la sentencia, combinamos texto con valores de las cajas de texto. Para hacer esto tenemos que anidar o concatenar los literales con el operador "&" y, además, rodear los TextBox con comillas simples, siempre y cuando lo que estamos buscando sean valores no numéricos.

El resultado de esta sentencia sería "Select * from Stock where Nombre='Calipo' and sabor='Menta'", en el caso en que pusiéramos estos valores (Calipo y menta). Esta búsqueda nos devolverá verdadero o falso en la propiedad "EOF" del Recordset.

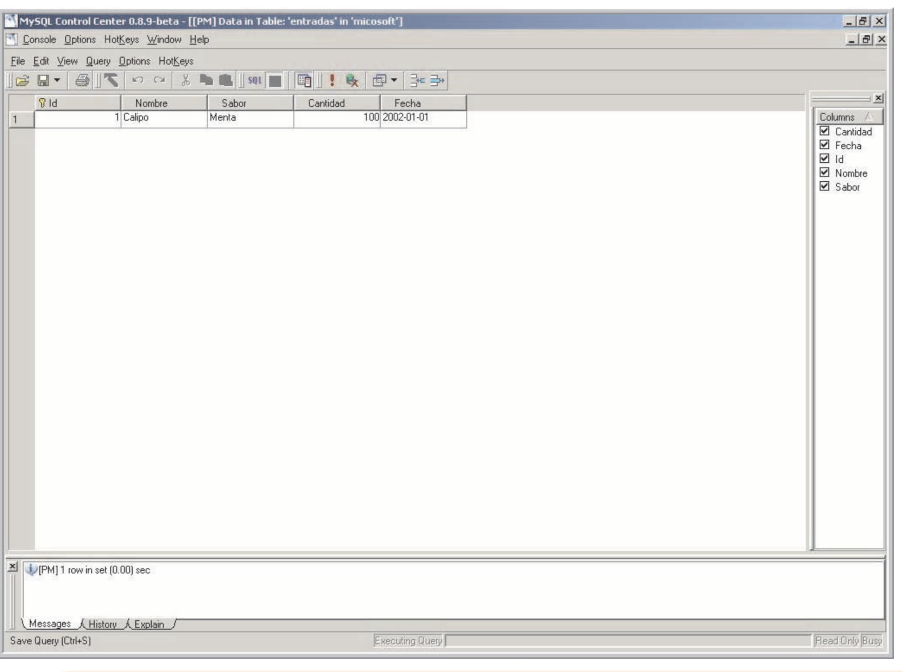
Posteriormente comprobamos si se ha encontrado algún registro con estos valores, y de ser así, en vez de dar de alta, modificamos el campo cantidad, avisando con un mensaje en pantalla. Y con esto podemos dar por finalizado, de momento, el formulario Stock.

Vamos a diseñar la tabla de entradas. Será idéntica a la tabla de Stock, más un campo de tipo Fecha que nos dirá que día fue incluida esa entrada. Para que nos entendamos, las entradas serán cajas de helados que nos traen de otras fábricas y están pendientes de validación para dar de alta en el stock. Por lo tanto, la tabla quedará así.

Ahora vamos al formulario. Será básicamente igual al de stock, más otra caja de texto que añadiremos para la fecha, aunque esta, en un principio, vendrá dada por el sistema. Este sería un posible diseño:



Introduzcamos un registro o dos en la tabla de entradas desde el administrador de MySql que estemos usando. Esto se hace haciendo doble click sobre la tabla en el MySql Control Central y, con el botón derecho, insertar nueva fila. Por ejemplo, yo he vuelto a añadir uno con Calipos de menta.



El código que hay entre el formulario de Stock y este no es muy diferente. Veamos que casi es un Copy – Paste del mismo, con algunas variaciones, claro. Este sería el Form_Load

```
Private Sub Form_Load()
    Set Rs = New ADODB.Recordset
    Rs.Open "entradas", Conn.ConnectionString,
    adOpenDynamic, adLockOptimistic
    While Not Rs.EOF
        With Lista.ListItems.Add(, , Rs("Id"))
            .SubItems(1) = Rs("Nombre")
            .SubItems(2) = Rs("Sabor")
            .SubItems(3) = Rs("Cantidad")
            .SubItems(4) = Rs("Fecha")
        End With
        Rs.MoveNext
    Wend
    MoverDatosACampos
    Deshabilitar
End Sub
```

Es básicamente igual al anterior, con la peculiaridad que ahora tenemos una nueva columna (Fecha), que debemos incluir en la lista. En el botón “Nuevo”, haremos que se cargue la fecha del sistema por defecto, permitiendo que el usuario pueda cambiarla.

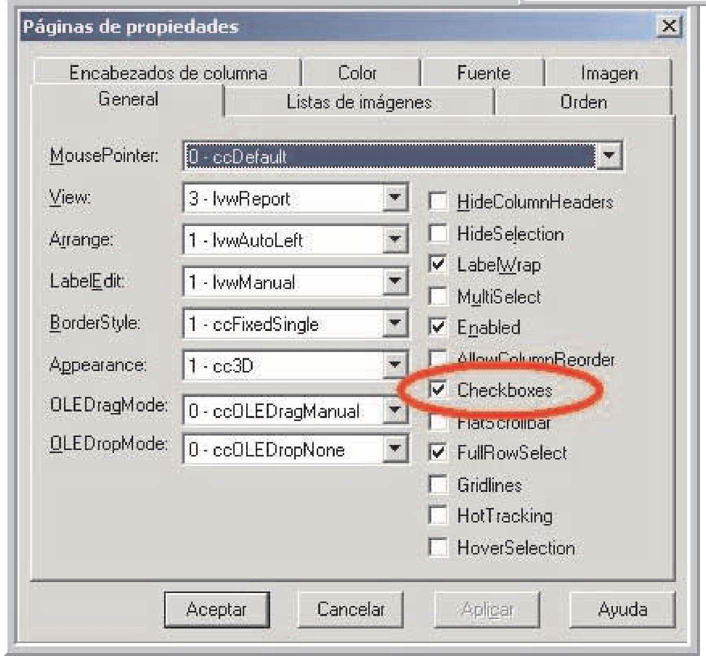
```
Private Sub CmdNuevo_Click()
    Habilitar
    Estado = "NUEVO"
    TxtFecha.Text = Date
    DeshabilitarBotones
    LimpiarCampos
End Sub
```

Lo único que tenemos que hacer para esto es introducir en la caja de texto “Fecha” el valor de la función “Date”.

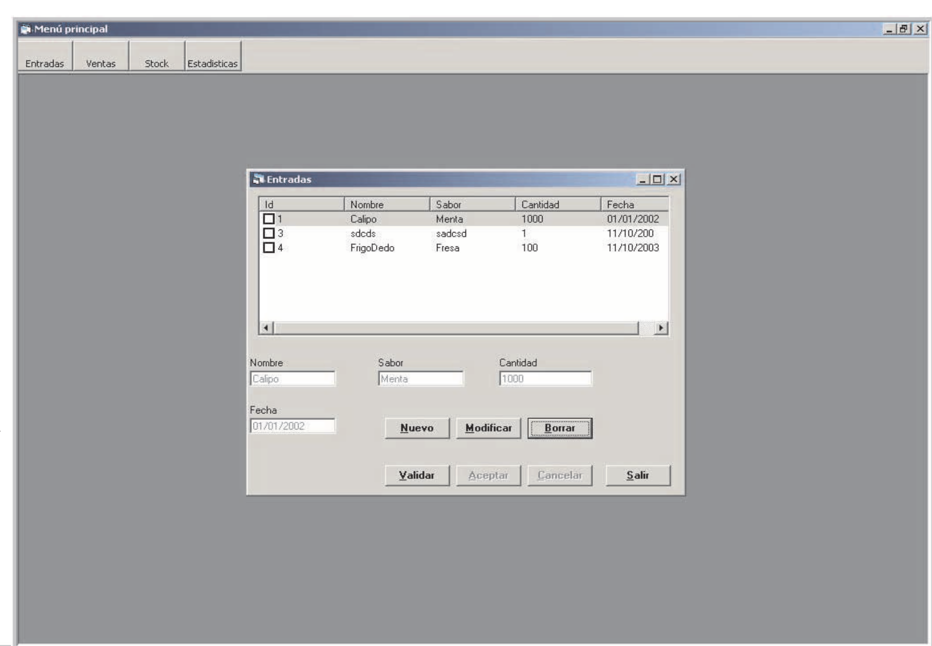
También tenemos unas pequeñas variaciones en las rutinas para habilitar y deshabilitar los campos de texto, obviamente debemos tratar nuestro nuevo objeto.

Sub Deshabilitar()
TxtFecha.Enabled = False
…………………
End Sub
Sub Habilitar()
TxtFecha.Enabled = True
…………………
End Sub

He puesto puntos suspensivos para no rescribir todo el código otra vez. Volvamos un momento a la lista. Pulsemos sobre el botón "Personalizado" y activemos la casilla "CheckBoxes" de la primera pestaña.



Aceptamos y vemos que en la primera columna nos aparece una casilla del tipo check. Si ejecutamos el programa apreciaremos mejor como nos aparece cada uno de los registros con un campo de tipo check. ¿Por qué lo hemos puesto? Os preguntareis. Pues porque ahora vamos a añadir un nuevo botón, el cual llamaremos "Validar", y que nos introducirá en el Stock todos aquellos registros que seleccionemos en este list.



Añadimos dos líneas de código nuevas en las rutinas de habilitar y deshabilitar botones.

Sub DeshabilitarBotones()
CmdValidar.Enabled = False
… … … … … …
End Sub

Sub HabilitarBotones()
CmdValidar.Enabled = True
… … … … … …
End Sub

Como vamos a utilizar checks para la selección de los registros sobre los que queremos actuar tenemos que hacer modificaciones en el botón aceptar, ya que ahora necesitaremos un bucle que recorra todas las entradas visibles en la lista y actuar contra las seleccionadas. Para ello yo voy a utilizar un bucle de tipo "FOR", ya que es el más indicado para este tipo de operaciones. Sería absurdo aplicar esto sobre una modificación de entrada, ya que solo se puede modificar un registro a la vez, pero talvez si estaría hacerlo en la eliminación, ya que podríamos borrar varias entradas de golpe.

Este sería el código:

```
Option Explicit
Dim Estado As String
Dim i As Integer

    Case "BORRAR"
      For i = 1 To Lista.ListItems.Count - 1
        If Lista.ListItems(i).Checked Then
          Rs.MoveFirst
          Rs.Find "Id=" & Lista.ListItems(i).Text
          If Not Rs.EOF Then
            Rs.Delete
          Else
            MsgBox "No se ha encontrado el registro seleccionado, se va a recargar
                  la lista", vbCritical, "Error"
            Call Form_Load
          End If
        End If
      Next
      Lista.ListItems.Clear
      Call Form_Load
      MsgBox "Registro borrado"
```

Expliquémoslo:
En caso de que se seleccione la opción borrar, abrimos un bucle de tipo "FOR". Anteriormente hemos declarado una variable de tipo Integer para recorrer los objetos de la lista.

Vemos el bucle: For i = 1 To Lista.ListItems.Count – 1
Le estamos diciendo que se repita desde la primera posición hasta la última, que sería el .Count – 1, porque el .Count sería uno después, y nos daría un error. A su vez, este bucle incrementa automáticamente la variable "i" en 1, para que podamos referirnos cada vez a un registro de la lista diferente.

Justo después vemos una sentencia condicional que pregunta si el registro "i" (sabemos que i se va incrementando hasta el total de filas en la lista) esta "cheked". Si es así, nos posicionamos al principio y buscamos, con el método Find, el registro correspondiente en la base de datos.

El resto es igual, preventivamente decimos con otra sentencia condicional que si la propiedad EOF del Recordset no es falso, lo borre con el método Delete. Para acabar limpiamos y recargamos la lista, mostrando un mensaje de conformidad.

Ahora vamos al botón más interesante de este formulario, el "Validar". Aquí tenemos una mezcla de conceptos. Debemos crear un bucle igual al anteriormente explicado, para recorrer una a una las filas de la lista. Una vez estemos posicionados en una de estas filas, debemos abrir una conexión contra la tabla stock, ya que tenemos que comprobar si el helado existe o no, porque de ser así, debemos sumar las cantidades y no dar de alta uno nuevo.

Este sería el código, que pasamos a comentar.

```
Option Explicit
Dim Estado As String
Dim i As Integer
Dim RsAlta As ADODB.Recordset
Private Sub CmdValidar_Click()
  For i = 1 To Lista.ListItems.Count - 1
    If Lista.ListItems(i).Checked Then
      Rs.MoveFirst
      Rs.Find "Id=" & Lista.ListItems(i).Text
      If Not Rs.EOF Then
        Set RsAlta = New ADODB.Recordset
        RsAlta.Open "Select * from stock where Nombre='" & Rs("Nombre")
        & "' and Sabor='" & Rs("Sabor") & "'", Conn.ConnectionString,
        adOpenDynamic, adLockOptimistic
        If Not RsAlta.EOF Then
          RsAlta("Cantidad") = RsAlta("Cantidad") + Rs("Cantidad")
          RsAlta.Update
        Else
          RsAlta.AddNew
          RsAlta("Nombre") = Rs("Nombre")
          RsAlta("Sabor") = Rs("Sabor")
          RsAlta("Cantidad") = Rs("Cantidad")
          RsAlta.Update
        End If
        RsAlta.Close
        Rs.Delete
      End If
    End If
  Next
  Lista.ListItems.Clear
  Call Form_Load
End Sub
```

En el punto en el que nos encontramos, no debería tener ninguna dificultad entender estas líneas. Sencillamente creamos el bucle "FOR", comprobando posteriormente si el registro de la lista está seleccionado.

Lo que le precede es sencillo. Buscamos el registro correspondiente en la tabla de entradas, mediante el método Find, Rs.Find "Id=" & Lista.ListItems(i).Text . Inmediatamente después, y si la propiedad EOF del Recordset no es verdadero, instanciamos un nuevo Recordset y lo abrimos, realizando una búsqueda sobre la tabla de stock por los campos Nombre y Sabor. En el caso en que la propiedad EOF del Recordset sea verdadero, daremos de alta un nuevo helado en la tabla stock, con la propiedad AddNew. Si por lo contrario se ha encontrado algún registro en la tabla stock que cumpla las condiciones, lo que haremos es sumar las cantidades de estos.

Para acabar, limpiamos la lista de objetos y llamamos al Form_Load para su recarga. Y con esto, supongo que podemos dar por finalizado el formulario de Entrada de productos.

Bien, para acabar este número creo que estaría bien mandaros unos "deberes" para casa. Para ser más exactos, creo que podríais intentar hacer la parte del proyecto que comprende las Ventas.

Daré una explicación del mismo:
El formulario de Ventas debe mostrar por pantalla una lista con las ventas de helados que los comerciales han ido realizando. Estas ventas deben incluir los campos Nombre, Sabor, Cantidad (obvios) y Fecha de venta. Cuando un comercial introduce una venta, este ve un formulario con los campos de entrada descritos anteriormente. Se debe poder añadir, modificar y borrar ventas, pero ninguna de estas ventas serán correctas hasta que sean validadas por un responsable de administración. Para ello añadiremos un botón Validar (al igual que en Entradas) que nos permitirá seleccionar aquellas ventas que queramos validar.

Una vez validadas, estos registros deben ser borrados de la tabla "ventas" y sus cantidades deben ser restadas del stock inicial. En el caso que algunas de las ventas realizadas no correspondan con ningún registro del stock, ya sea porque en ese momento no hubiera, o porque se ha vendido un helado que no existe, este deberá quedar marcado de alguna manera, pero nunca debe repercutir sobre la tabla de stock. Buena suerte!!!

Nota: Tienes el cógigo de este artículo en www.hackxcrack.com

PROGRAMACION EN GNU LINUX
DESARROLLO DE APLICACIONES EN ENTORNOS UNIX E INICIACIAON AL LENGUAJE C (III)

el_chaman. LUIS U. RODRIGUEZ PANIAGUA

Mes a mes avanzamos en la programación en C desde LINUX Esta vez nos enfrentaremos a las estructuras, enumeraciones, uniones y campos de bits.
Nos acercaremos a la programación modular y crearemos nuestro primer programa "serio" :)

1. Introducción.

En el número anterior nos enfrentamos a los punteros. Para dar fin a la parte relativa a los tipos de datos, hoy veremos las estructuras, las enumeraciones, las uniones y los campos de bits. Así mismo veremos como a partir de los tipos vistos en C, podemos crear nuestros propios tipos (sencillos o como se verá en el ejemplo de este número, muy complejos) mediante **typedef**.

Así mismo en este número llevaremos el concepto de la programación modular al propio código fuente, no limitándonos ya tan sólo a repartir el código fuente entre distintos archivos, sino dividiendo éste en módulos independientes denominados procedimientos o funciones.

2. Tipos de datos en C, tipos derivados, continuación

2.1. Estructuras

Una estructura es un agrupamiento similar a un array con la diferencia de que cada una de las "casillas" que componen el array pueden ser de diferente tipo (ya sean tipos fundamentales o derivados). Su máxima utilidad radica en que en ocasiones nos conviene tener un conjunto de información dentro de un conjunto, ya sea porque esa información está relacionada de alguna forma, ya sea porque deseamos mantenerla junta.

Por ejemplo, imaginemos el caso de los datos del DNI de una persona. En un primer vistazo, se nos puede ocurrir representar los datos relativos a un DNI de la siguiente manera:

```
....
char nombre[30];
char ape1[30];
char ape2[30];
char sexo[3];
char estado_civil[15];
char calle[30];
int  cp[6];
char poblacion[30];
char dni[12];
....
```

Obviamente estos tipos pueden ser escogidos de manera diferente. Lo importante es que estemos de acuerdo en que todas estas variables

tienen algo en común: Son información relativa al DNI de **una** persona.

¿Qué ocurriría si deseásemos representar los datos de los DNIs de dos personas? Probablemente obtendríamos algo como esto:

```
....
char nombre_1[30];
char ape1_1[30];
char ape2_1[30];
char sexo_1[3];
char estado_civil_1[15];
char calle_1[30];
int  cp_1[6];
char poblacion_1[30];
char dni_1[12];
char nombre_2[30];
char ape1_2[30];
char ape2_2[30];
char sexo_2[3];
char estado_civil_2[15];
char calle_2[30];
int  cp_2[6];
char poblacion_2[30];
char dni_2[12];
```

Si ya nos planteamos tratar con los datos relativos al DNI de diez o quinientas personas, vemos que hay algo "que no cuadra" o que al menos se debería hacer de otra manera.

Será entonces cuando cobre sentido las estructuras. Veamos primero un ejemplo de estructura y luego observemos las diferencias con los expuestos arriba:

```
....
struct DNI{
  char nombre[30];
  char ape1[30];
  char ape2[30];
  char sexo[3];
  char estado_civil[15];
  char calle[30];
  int  cp[6];
  char poblacion[30];
  char dni[12];
};
....
main()
{
   struct DNI persona1, persona2, persona3, persona4, persona5;
   ......
```

Podemos observar que hemos "empaquetado" toda la información relativa a un dni en un "conjunto" llamado DNI. Posteriormente en la zona de declaración de variables creamos cinco variables cuyo tipo es... ¡el conjunto que hemos creado!

Esto proporciona dos ventajas: Por un lado no tenemos que manejar una cantidad terrible de variables y por el otro será más difícil el "traspapelar" información entre los dnis de dos personas distintas.

Bueno, a estas alturas seguro que estamos convencidos de las bondades de las estructuras, pero probablemente alguien piense, y no sin razón: "¿Cómo demonios accedo ahora a las variables empaquetadas para asignarles un valor o consultar el que tienen?".

Pues el mecanismo será el siguiente:
nombre_estructura.nombre_variable

Por ejemplo:

```
scanf("%s", persona1.nombre);
persona1.cp=47089;
printf("%s %i", persona1.nombre,persona1.cp);
```

Obsérvese que los datos nombre y cp han sido modificados/consultados como si de una variable normal se tratase (atendiendo al tipo original de nombre y cp). Ahora bien, sólo hemos trabajado con los datos nombre y cp de persona1 sin que se hayan visto alterados los mismos campos de persona2, persona3, etc.... dado que son conjuntos distintos de datos.

Estructuras y punteros

Una estructura puede ser apuntada mediante un puntero. Esto provocará ciertos cambios en su comportamiento que debemos tener muy en cuenta. Al igual que arriba, veamos esto con un ejemplo:

```
....
struct DNI{
  char nombre[30];
  char ape1[30];
  char ape2[30];
  char sexo[3];
  char estado_civil[15];
  char calle[30];
  int  cp[6];
  char poblacion[30];
  char dni[12];
};
....
main()
{
  struct DNI persona1, *puntero_dni;
  ......
```

Mientras que hemos visto que para acceder a las distintas variables que forman parte de la estructura tenemos que usar una sintáxis del tipo (según el ejemplo) persona1.nombre_variable, si tenemos un puntero esto cambiará un poco:

```
/* Así funciona lo visto hasta ahora */
printf("Deme el nombre de una persona: ");
scanf("%s", persona1.nombre);
printf("%s", persona1.nombre);
/* Comenzamos a usar el puntero */
/* Inicializamos el valor del puntero: ¡ ES UN PUNTERO ! */
puntero_dni = &persona1;
/* Accedemos al dato nombre */
printf("%s", puntero_dni->nombre;
```

¿Qué ha sucedido? Pues ha sucedido que cuando una estructura es referenciada por un puntero, para acceder a sus campos debemos emplear al símbolo -> en vez de el punto que empleamos cuando simplemente estamos trabajando con una variable de tipo estructura de datos.

Es algo que deberemos tener siempre en cuenta.

Finalmente, y para que quede constancia, debemos de insistir en una cosa: Pueden formar parte de una estructura cualquier conjunto de variables de cualquier tipo. Esto quiere decir que nos podremos encontrar estructuras tan exóticas como:

```
struct fecha{
    unsigned int dia;
    unsigned int mes;
    unsigned int anyo;
};
struct libro{
  char titulo[40];
  char autor[40];
  char ISBN[15];
  int  codigo;
};
struct socio{
  char nombre[30];
  char apellidos[60];
  int  num_socio;
  int num_multas;
```

```
    struct fecha multas[5];
};
struct prestamo{
    struct fecha inicio;
    struct fecha fin;
    struct libro *libros_prestados;
    struct socio datos_socio;
};
```

Se obvia que los tipos que faltan por tratar también pueden formar parte de una estructura.

También veremos, en un tema aparte, que las estructuras se pueden autoreferenciar o referenciar a otras estructuras. Esto nos permitirá realizar estructuras aún más complejas como listas dinámicamente enlazadas. Pero debido a la complejidad de este tema merece un artículo por separado.

Finalmente, propongo como ejercicio que se realice un sencillo programa en C en el que se preste al usuario Pepe "La divina Comedia" de Dante Alighieri y que se imprima TODA la información relativa a dicho préstamo bajo los siguientes supuestos: El usuario Pepe sufrió una multa el 01/01/01 y además posee actualmente en préstamo "El derecho a leer" de Richard M. Stallman.

2.2. Enumeraciones

Las variables de tipo enumeración son un tipo especial de variables que poseen la propiedad de que su conjunto de valores en un conjunto de constantes enteras especificadas por el usuario. Para mayor comprensión, estos valores enteros se especifican mediante términos comunes.

Así por ejemplo serán enumeraciones las siguientes:

```
enum dias    {lunes, martes, miercoles, jueves, viernes, sabado, domingo};
enum colores { rojo, azul, verde};
.....
main(){
        enum dias     dia_laborable;
        enum colores  color_tejado;
        dia_laborable=lunes;
        color_tejado=azul;
.....
```

Aunque parezca mentira estamos tratando todo el rato con enteros (0, 1, 2, 3, 4, 5 y 6 para dias y 0, 1 y 2 para colores), pero de manera que en vez de utilizar las cifras, usamos los distintos elementos que hemos asignado al conjunto.

```
#include <stdio.h>
enum dias { lunes, martes, miercoles, jueves, viernes, sabado, domingo};
main()
{
  enum dias laborable;
  laborable=martes;
  if (laborable<=viernes)
        {
        printf(" Quedan %i para el fin de semana ", sabado - laborable);
        }
  else
        {
        printf("¿Qué haces trabajando en fin de semana loco?");
        }
  }
```

Un fallo o tentación común es esperar que se nos imprima el término que hemos empleado para designar uno de los números. Por ejemplo:

```
laborable = martes;
printf(" Hoy es %s ", laborable);
```

Esto será **INCORRECTO** dado, y aprovechamos para recordarlo una vez más, una enumeración no es más que un conjunto de números enteros.

Una posible solución para esto será emplear un código como el siguiente:

```
color=rojo;
if(color==rojo)
printf("El color es el rojo");
if(color==negro)
printf("El color es el negro");
...
```

Aunque a primera vista esto pueda parecer engorroso, pronto veremos que el uso de las funciones nos facilitará bastante esta tarea.

2.3. Uniones

Las uniones se definirán de manera similar a las estructuras y se emplean para almacenar en un mismo espacio de memoria variables de distintos tipos. Recordemos que el concepto de tipo visto en números anteriores era "el número de casillas que reservábamos en memoria para almacenar datos". Vimos que había tipos que ocupaban muchas casillas y otros menos. Pues bien, en una unión sólo reservaremos las casillas del elemento que más ocupe en la unión, almacenándose el resto de los elementos dentro de las casillas del elemento más grande (dado "que caben").

Veamos esto con un ejemplo:

```
union numero
    {
      float real;
      int   entero;
    };
    ....
    main()
    {
      union numero un_numero;
      un_numero.entero=5;
      un_numero.real=1555;
    }
```

Obviamente en el ejemplo mostrado **un_numero.entero** deja de valer 5 tras asignar **un_numero.real=1555** dado que ambos números utilizan las mismas zonas de memoria para almacenarse y por lo tanto se sobrescriben.

2.4. Campos de bits

El lenguaje C nos brinda la posibilidad de definir variables cuyo tamaño en bits pueda no coincidir con un múltiplo de 8 (byte). Un ejemplo de uso será:

```
struct mi_byte
  {
    unsigned char bit0:1; /* nombre:longitud en bits del campo */
    unsigned char bit1:1;
    unsigned char bit2:1;
    unsigned char bit3:1;
    unsigned char bit4:1;
    unsigned char bit5:1;
    unsigned char bit6:1;
    unsigned char bit7:1;
  };
```

Como podemos observar lo que hemos hecho ha sido "trocear" un char (1 byte) en los 8 bits que lo componen, de manera que ahora tenemos acceso a cada uno de los bits de manera independiente. Si utilizamos lo visto en el apartado anterior, podemos hacer algo como:

```
#include <stdio.h>
  struct mi_byte
  {
    unsigned char bit0:1;
   /* nombre:longitud en bits del campo */
    unsigned char bit1:1;
    unsigned char bit2:1;
    unsigned char bit3:1;
    unsigned char bit4:1;
    unsigned char bit5:1;
    unsigned char bit6:1;
    unsigned char bit7:1;
  };
  union un_byte
  {
    unsigned char   byte;
    struct mi_byte  bits;
  };
  main()
  {
   union un_byte big_endian, little_endian;
     little_endian.byte=0x3E;
             big_endian.bits.bit4=little_endian.bits.bit0;
             big_endian.bits.bit5=little_endian.bits.bit1;
```

```
        big_endian.bits.bit6=little_endian.bits.bit2;
        big_endian.bits.bit7=little_endian.bits.bit3;
        big_endian.bits.bit0=little_endian.bits.bit4;
        big_endian.bits.bit1=little_endian.bits.bit5;
        big_endian.bits.bit2=little_endian.bits.bit6;
        big_endian.bits.bit3=little_endian.bits.bit7;
        printf("LE:  %2x BE: %2x\n", little_endian, big_endian);
    }
```

En el ejemplo visto simulamos la resolución de un antiguo problema que existía entre diversas máquinas. Este problema consistía en que en algunos procesadores (por ejemplo, la familia 8088) guardaban los bytes poniendo el nibble (4 bits) menos significativos en primer lugar, y los más significativos en segundo. A esta manera de hacer las cosas se le conoce como Big Endian. Otras arquitecturas, como por ejemplo la basada en MC68000, lo hacían al revés (Little Endian). Esto en ocasiones provocaba que el intercambiar datos fuese una tarea ardua, dado que cada byte tenía que ser sometido al proceso mostrado en el ejemplo, en el cual lo único que hacemos es intercambiar el orden de los nibles adecuados. Otra aplicación posible de lo visto sería la construcción de un emulador entre procesadores incompatibles entre si a este nivel.

La salida del programa arriba mostrado será:

```
LE:  bffff93e BE: bffff9e3
```

Obsérvese que al último byte le hemos "dado la vuelta".

2.5. typedef: Poniendo nombre a nuestros propios tipos de datos

Hasta el momento hemos visto como con una serie de tipos datos derivados hemos sido capaces de crear nuestros propios tipos de datos: Para ello hemos empleado estructuras, enumeraciones, uniones, campos de bits.....

Pero queda una pequeña pega que más que un problema real es una cuestión estética. Observemos estas líneas de código:

```
main()
    {
      struct DNI     persona1, persona 2;
      enum   dias    dia_laborable, dia_festivo;
      union  numero  entero_o_real;
      int    n_entero;
      float  n_real;
      char   un_caracter;
    .....
    }
    main()
    {
      DNI      persona1, persona2;
      dias     dia_laborable, dia_festivo;
      numero   entero_o_real;
      int      n_entero;
      float    n_real;
      char     un_caracter;
    .....
    }
```

En el segundo main tenemos la impresión de que nuestros tipos "son más tipos" que arreglos que hemos tenido que hacer. Se comportan como el resto de los tipos fundamentales (int, float y char) y proporcionan una mayor sensación de homogeneidad en el código.

Ahora bien, ¿cómo se consigue que podamos utilizar los nombres de los tipos de la segunda maneara? La respuesta es mediante typedef. typedef hará posible que cualquier identificador que nosotros asignemos a un tipo derivado sea considerado como el nombre de un nuevo tipo. Así por ejemplo para conseguir el último resultado estaríamos a reescribir nuestro código de la siguiente manera:

```
  ....
    typedef struct DNI{
      char nombre[30];
      char ape1[30];
      char ape2[30];
      char sexo[3];
```

```
    char estado_civil[15];
    char calle[30];
    int  cp[6];
    char poblacion[30];
    char dni[12];
} DNI;


typedef enum dias { lunes, martes, miercoles, jueves, viernes, sabado, domingo} dias;


typedef union numero
{
    float real;
    int   entero;
}numero;

main()
{
    DNI      persona1, persona2;
    dias     dia_laborable, dia_festivo;
    numero   entero_o_real;
    int      n_entero;
    float    n_real;
    char     un_caracter;
.....
}
```

El uso de typedef no se limita a esto y puede ser empleado para cosas más triviales como:

```
.....
typedef float REAL;
typedef int   ENTERO;
typedef char  BYTE;
main()
{
   REAL     pi, hipotenusa;
   ENTERO   num_manzanas;
   BYTE     letra;
.....
}
```

3. Programación modular

3.1. Funciones

En números anteriores vimos como aplicar la modularidad a todo un proyecto dividiendo este en diferentes archivos según la funcionalidad que tuviese el código residente en estos archivos. Hoy vamos a meternos dentro del código y veremos que esta filosofía puede ser aplicada también dentro del propio código.

El mecanismo del que dispondremos para realizar esta tarea serán las funciones o procedimientos.

¿Qué será una función o procedimiento? Serán unas porciones de código que recibirán unos datos, realizarán una determinada tarea con ellos y nos proporcionarán un resultado. Estas porciones de código deben de estar adecuadamente identificadas, de manera que se las pueda llamar desde cualquier punto del programa principal.

Pongamos por ejemplo una tarea sencilla: sumar dos números. ¿Cómo podemos llamar a una función que sume dos números? Por ahora admitamos suma como un nombre válido. ¿Cuántos datos necesita conocer esta función para poder realizar su tarea? Obviamente los dos sumandos. A los datos que necesita una función para trabajar los llamaremos a partir de ahora parámetros. Y finalmente ¿cómo llamaremos al producto final del trabajo que realiza esta función? Pues... suma. Claro que veremos que este dato se le denomina **valor de retorno**.

Ajustándonos al estándar ANSI, el aspecto de una función debe de ser el siguiente:

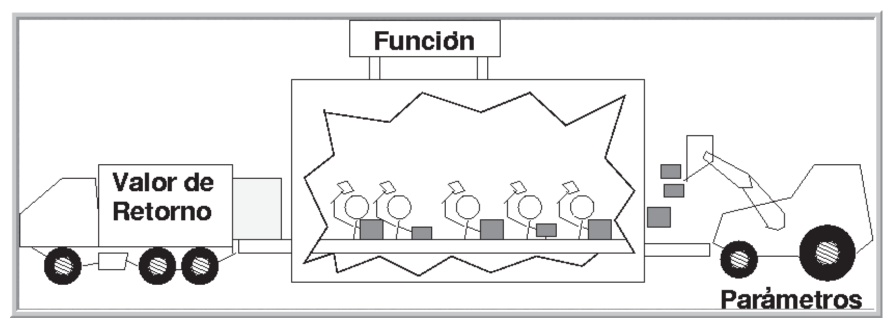
```
tipo_retorno  nombre_funcion(declaracion_parametros)
{
    variables_locales;
    instrucciones;
    return (valor de retorno);
}
```

Así nuestra función suma podría quedar en C:

```
#include <stdio.h>
 /* Declaración de funciones */
  /* Función que suma dos números y retorna o devuelve el resultado * de la suma*/
  float suma(float op1, float op2)
```

```
{
    float resultado;
    resultado=op1+op2;
    return resultado;
}
/* Función principal o punto de entrada del programa */
main()
{
    float a, b, c;
     printf("\n Deme un número: ");
    scanf("%f",&a);
    printf("\n Deme otro número: ");
    scanf("%f",&b);
    c=suma(a,b);
    prinf("\n El resultado de %4.2f + %4.2f es %5.2f", a, b, c);
    printf("\n Deme un número: ");
    scanf("%f",&a);
    printf("\n Deme otro número: ");
    scanf("%f",&b);
    c=suma(a,b);
    prinf("\n El resultado de %4.2f + %4.2f es %5.2f", a, b, c);
}
```

Podríamos observar que una función se comporta como una fábrica que recibe materias primas y proporciona un producto.



Ahora bien; no siempre una función se comportará de esta manera. De hecho existe una clasificación atendiendo a lo que recibe y a lo que devuelve. Esta clasificación considera que las funciones que no retornan ningún resultado (por ejemplo, imprimir un menú o un mensaje de error) deben de ser llamadas **procedimientos** mientras que las que sí retornan algún valor han de ser llamadas propiamente **funciones**.

El aspecto de un procedimiento suele ser el siguiente:

```
void imprime_menu(void)
    {
        printf("\n 1 - Añadir usuario");
        printf("\n 2 - Borrar usuario");
      printf("\n 3 - Salir");
    }
```

En general, tomando el esqueleto de una función tendremos que:

```
tipo_valor_retorno    nombre_funcion( tipo param1, tipo param2,...., tipo paramN)
    {
        tipo var_local1;
        tipo var_local2;
        ....
        tipo var_localJ
        instrucciones;
        return valor;
    }
```

Aquí debemos de llamar la atención sobre varias cosas:

La primera es el tratamiento de las variables locales. Esta variable **sólo existe durante la ejecución de la función.** Esto quiere decir que una vez que termina la función éstas desaparecen. Este tipo de variables tampoco son visibles desde fuera de la función. Dicho de otra manera: Son herramientas que sólo los trabajadores de la fábrica pueden manejar. Estas herramientas no se pueden utilizar desde fuera y ellas no pueden realizar ninguna actividad fuera de los muros de la fábrica.

La segunda es qué ocurre cuando nuestra función es un procedimiento; es decir, cuando no retorna nada. En ese caso se pone como tipo del valor de retorno void lo cual significa precisamente "nada" [1]. Además dentro del procedimiento se prescinde también de la instrucción return dado que ahora carece de sentido su uso. *(1) Esto no es del todo cierto, ya que como veremos un puntero *void significa literalmente que es un puntero que puede apuntarlo a todo.*

Cuando no se reciben parámetros, se suele poner también un void entre los paréntesis u omitirlo:

```
float sumar(float a, float b)
{
   return a + b;
}
void imprime_error(int codigo)
{
  if(codigo==1)
  {
    printf("\n Error chungo\n")
  }
  else
  {
    printf("\n Error desconocido");
  }
}
void imprime_menu(void)
{
  printf("\n 1 - Añadir usuario");
  printf("\n 2 - Borrar usuario");
  printf("\n 3 - Salir");
}
void logo()
{
  printf("\n (c) el_chaman 2003\n");
}
```

Una función podrá retorna cualquier tipo de valor, salvo un array u otra función.

Como se explicó anteriormente, las variables locales a una función permanecen sólo durante la ejecución de esta. Esto sucede porque dichas variables se almacenan en la pila del usuario de programa. También se las denomina variables automáticas. Sin embargo podemos hacer que una variable local sea almacenada como una global y no en la pila, existiendo a lo largo de toda la ejecución del programa.

A este tipo de variables se les denomina estáticas, y se caracterizan porque llevan la palabra static delante de la declaración de la variable.

3.1.1. Parámetros por valor y parámetros por referencia

Tal como hemos utilizado los parámetros hasta ahora sucede algo como lo siguiente:

```
.....
float sumar(float a, float b)
{
  return a+b;
}
main()
{
  float numA, numB, resultado;
  numA=34.67;
  numB=70;
  resultado=sumar(numA, numB);   (1)
.....
```

Lo que sucede en la línea **(1)** es que los parámetros reciben una copia de las variables que se le pasan. Es decir, que para uso de la función hacemos algo como a=numA, b=numB. A esta manera de hacer las cosas se le llama **pasar parámetros por valor**.

Otro comportamiento de los parámetros pasados por valor es que aunque cambiemos el valor de los mismos dentro de una función, este cambio no se verá reflejado fuera de la función. Veamos esto con un ejemplo.

```
.....
void intercambia(float a, float b)
{
  float tmp;
  tmp = a;
  a=b;
```

```
    b=tmp;
  }
 main()
  {
     float a, b;
     a=1;
     b=2;
    intercambia(a,b); (1)
    printf("a:%f, b:%f\n", a, b);
  }
```

La intención que tenemos a la hora de construir la función (procedimiento) intercambia es que nos intercambie los dos valores que se nos pasen como parámetros. Sin embargo esto no sucede. Vemos por qué.

Al realizar la llamada (1), sucede lo siguiente:

```
intercambia(a(local)=a(global), b(local)=b(global))
  {
     tmp(local)=a(local);
     a(local)=b(local);
     b(local)=tmp(local);
  }
  /* Desaparecen a(local), b(local) y tmp(local) */
```

Bueno, vemos que el lugar donde hemos hecho los cambios se desvanece nada más terminar la función, luego las variables globales no se ven afectadas.

La conclusión que podemos sacar de esto es que utilizaremos el paso de parámetros por valor, cuando queramos pasar una copia de los datos a la función sin que éstos se vean modificados por la misma.

Sin embargo, en ejemplos como el arriba expuesto, o cuando una función necesite retornar más de un valor, nos conviene pasar determinados parámetros **por referencia**.

Veamos ahora una nueva versión del ejemplo arriba mostrado:

```
.....
  void intercambia(float *a, float *b)
  {
    float tmp;
    tmp =*a;
    *a=*b;
    *b=tmp;
  }
  main()
  {
    float a, b;
    a=1;
    b=2;
    intercambia(&a,&b); (1)
   printf("a:%f, b:%f\n", a, b);
  }
```

Si ejecutamos ahora este programa veremos que sí se han intercambiado los valores de las variables globales a y b. ¿Qué ha sucedido?

De entrada podemos observar que se han realizado algunos cambios en el código. El principal es que en los parámetros hemos sustituido la declaración "normal" por declaración de... ¡punteros!

```
void intercambia(float *a, float *b)
```

Esto quiere decir que esta función ahora NO recibirá una copia de los datos que deseemos sino **la dirección de la variable que contiene el dato original**. Esto querrá decir que todos los cambios que hagamos en una dirección global desde un puntero permanecerán tras la ejecución de la función.

Veámoslo gráficamente:

Parámetros por valor

Mem Global: a = 1, b = 2 — Pila: (vacía)

```
main()
{
   float a,b;

   a=1;
   b=2;
   .....
```

Mem Global: a = 1, b = 2 — Pila: a = 1, b = 2, tmp = ?

```
.....
intercambia(a,b);
.....
```

Mem Global: a = 1, b = 2 — Pila: a = 2, b = 1, tmp = 1

```
{
   tmp = a;

   a = b;
   b = tmp;

}
```

Mem Global: a = 1, b = 2 — Pila: (vacía)

```
/* Fin del programa */

 printf(" a:%f b:%f \n", a, b);
}
```

Parámetros por referencia

Mem Global: a = 1, b = 2 — Pila: (vacía)

```
main()
{
   float a,b;

   a=1;
   b=2;
   .....
```

Mem Global: a = 1, b = 2 — Pila: *a =, *b =, tmp = ?

```
.....
intercambia(&a, &b);
.....
```

```
/* Llamada a funcion */
```

Mem Global: a = 2, b = 1 — Pila: *a =, *b =, tmp=1

```
{
   tmp = *a;

   *a = *b;
   *b = tmp;

}
```

Mem Global: a = 2, b = 1 — Pila: (vacía)

```
/* Fin del programa */

 printf(" a:%f b:%f \n", a, b);
}
```

3.2. Poniendo en práctica lo aprendido

Tras varios artículos de teoría más o menos densa pero que no podía ser obviada, va siendo hora de que empecemos a realizar algún programa medianamente serio.

Para comenzar vamos a realizar la implementación de un **Tipo Abstracto de Datos (T.A.D.)** correspondiente a una matriz bidimensional. El tratamiento de T.A.D.s es tan profundo que incluso cubre el temario de alguna asignatura de Ingeniería Informática, por lo que no profundizaremos demasiado en el concepto. Digamos por ahora que un T.A.D. nos sirve para definir tanto un tipo de datos como todos los mecanismos que nos permitirán acceder a los datos almacenados en él.

En concreto vamos a construir una matriz en

a que podamos meter cualquier tipo de dato y esta pueda ser de cualquier tamaño (que no dimensión).

Debido a que he procurado comentar el código extensamente, prescindiré de mayores explicaciones en este artículo. Siempre que se tengan dudas disponéis del foro o de mi dirección de correo: luisb@es.gnu.org

```
/*
 * ARCHIVO: matriz.h
 *
 * DESCRIPCIÓN:
 *  Este archivo es el archivo de cabecera correspondiente al T.A.D. matriz.
 *  En él vamos a declarar los prototipos de las funciones así como las
 *  estructuras necesarias para implementar dicho T.A.D.
 *
 * AUTOR: Luis U. Rodríguez Paniagua
 * Este archivo se distribuye bajo los términos de la licencia GNU. Para más
 * información, consultar el archivo
 * LICENCIA.pdf o http://es.tldp.org/Otros/gples/gples.html
 *
 */
#ifndef _MATRIZ_H_
#define _MATRIZ_H_
#include <stdlib.h>
#include <stdio.h>

/* Definimos una macro (código que sustituirá POS(pM, i, j) por
 * (char *) pM->elementos + pM->tamagno * ((i*pM->n) + j)
 * cada vez que lo encuentre en el código ) con el fin de transformar unas
 * coordenadas del tipo i, j en un desplazamiento sobre una dirección de
 * memoria base referida al puntero pM->elementos
 *
 */
#define POS(pM, i, j) ( (char *) pM->elementos + pM->tamagno * ((i*pM->n) + j))

/* Estructura de datos que definirá el T.A.D. matriz */

typedef struct Matriz{
	int   m;          /* Filas     */
	int   n;          /* Columnas */
	int   tamagno;    /* Tamaño en bytes de la matriz */
	void *elementos;  /* Puntero a los datos de la matriz */
}Matriz;

/* Procedimiento encargado de crear una nueva matriz */
void MatrizCrea(Matriz *pM, int m, int n, int tamagno);

/* Procedimiento que destruye una matriz existente */
void MatrizDestruye(Matriz *pM);

/* Procedimiento que obtiene el componente i,j-ésimo de la matriz */
void MatrizObten(Matriz *pM, int i, int j, void *elemento);

/* Procedimiento que deposita un dato en la componente i.j-ésima de la matriz */
void MatrizCambia(Matriz *pM, int i, int j, void *elemento);

#endif
/* Fin del archivo matriz.h */
```

Este será el archivo de cabecera. En el ponemos las estructuras, variables, constates, macros, etc... que estén relacionadas con la matriz. Así mismo ponemos los prototipos (declaraciones) de las funciones que serán usadas. La implementación de estas funciones irán en un archivo aparte (matriz.c).

Obsérvese que un archivo de cabecera comienza y termina siempre por

```
#ifndef _LOQUESEA_
    #define _LOQUESEA_
    ....
    ....
   #endif
```

Esto se utiliza para el caso de que accidentalmente carguemos un archivo de cabecera más de una vez (cosa muy probable dado que nunca sabemos que archivos de cabecera cargan los archivos de cabecera que cargamos nosotros). De esta manera en el caso de que se cargase dos veces el mismo archivo de cabecera, la segunda vez lo ignoraría.

El archivo matriz.c contendrá:

```
/*
 * ARCHIVO: matriz.c
 *
 * DESCRIPCIÓN:
 *  Este archivo es el archivo .c que acompaña al archivo de cabecera matriz.h
 *  En él definimos las funciones cuyos prototipos se escrivieron en matriz.h
```

```
 * AUTOR: Luis U. Rodríguez Paniagua
 * Este archivo se distribuye bajo los términos de la licencia GNU. Para más
 * información, consultar el archivo
 * LICENCIA.pdf o http://es.tldp.org/Otros/gples/gples.html
 *
 */
#include "matriz.h"

/* Procedimiento encargado de crear una nueva matriz */
void MatrizCrea(Matriz *pM, int m, int n, int tamagno)
{
	pM->m = m;
	pM->n = n;
	pM->tamagno = tamagno;
	pM->elementos = calloc(m*n, tamagno);
}

/* Procedimiento que destruye una matriz existente */
void MatrizDestruye(Matriz *pM)
{
	free(pM->elementos);
}

/* Procedimiento que obtiene el componente i,j-ésimo de la matriz */
void MatrizObten(Matriz *pM, int i, int j, void *elemento)
{
	memcpy(elemento, POS(pM,i,j), pM->tamagno);
}

/* Procedimiento que deposita un dato en la componente i,j-ésima de la matriz */
void MatrizCambia(Matriz *pM, int i, int j, void *elemento)
{
	memcpy(POS(pM,i,j), elemento,pM->tamagno);
}
/* Fin del archivo matriz.c */
```

Ahora, además, vamos a crearnos otros archivos (dni.h y dni.c) para manejar información relativa a DNIs:

```
/*
 * ARCHIVO: dni.h
 *
 * DESCRIPCIÓN:
 *  Este archivo contiene las estructuras de datos y los prototipos de las
 *  funciones necesarios para manejarlas relativas a un DNI
 *
 * AUTOR: Luis U. Rodríguez Paniagua
 *
 * LICENCIA:
 * Este archivo se distribuye bajo los términos de la licencia GNU. Para más
 * información, consultar el archivo
 * LICENCIA.pdf o http://es.tldp.org/Otros/gples/gples.html
 *
 */
#ifndef _DNI_H_
#define _DNI_H_
#include <string.h>
#include <stdlib.h>
#include <stdio.h>

typedef struct DNI{
	char nombre[30];
	char ape1[30];
	char ape2[30];
	char sexo[3];
	char estado_civil[15];
	char calle[30];
	char cp[6];
	char poblacion[30];
	char dni[12];
	/*char *foto; Futuras mejoras */
}DNI;
/* Función encargada de meter datos en un DNI */
void pon_datos_dni(char *nombre, char *ape1, char *ape2, char
*sexo, char *estado_civil, char *calle, char *cp, char *poblacion,
char *dni, DNI *un_dni);
```

```
/* Función encargada de imprimir los datos de un DNI */
void imprime_dni(DNI un_dni);
#endif
/*
 * ARCHIVO: dni.c
 *
 * DESCRIPCIÓN:
 *  Este archivo contiene las definiciones correspondientes a las funciones
 *  cuyos prototipos se encuentran en dni.h
 *
 * AUTOR: Luis U. Rodríguez Paniagua
 *
 * LICENCIA:
 * Este archivo se distribuye bajo los términos de la licencia GNU. Para más
 * información, consultar el archivo
 * LICENCIA.pdf o http://es.tldp.org/Otros/gples/gples.html
 *
 */
#include "dni.h"

/* Función encargada de meter datos en un DNI */

void pon_datos_dni(char *nombre, char *ape1, char *ape2, char
*sexo, char *estado_civil, char *calle, char *cp, char *poblacion,
char *dni, DNI *un_dni)
{
       strcpy(un_dni->nombre, nombre);
       strcpy(un_dni->ape1, ape1);
       strcpy(un_dni->ape2, ape2);
       strcpy(un_dni->sexo, sexo);
       strcpy(un_dni->estado_civil, estado_civil);
       strcpy(un_dni->calle, calle);
       strcpy(un_dni->cp, cp);
       strcpy(un_dni->poblacion, poblacion);
       strcpy(un_dni->dni, dni);
}

/* Función encargada de imprimir los datos de un DNI */
void imprime_dni(DNI un_dni)
{
       printf("\n DNI: %s ", un_dni.dni);
       printf("\n Nombre: %s ", un_dni.nombre);
       printf("\n Apellidos: %s %s", un_dni.ape1, un_dni.ape2);
       printf("\n Calle: %s, %s, %s", un_dni.calle, un_dni.cp, un_dni.poblacion);
       printf("\n S: %s   E.C.: %s \n", un_dni.sexo, un_dni.estado_civil);
}
```

Y finalmente un programa principal que utilice todo lo hecho anteriormente:

```
/*
 * ARCHIVO: ppal.c
 *
 * DESCRIPCIÓN:
 *  Programa principal
 *
 * AUTOR: Luis U. Rodríguez Paniagua
 *
 * LICENCIA:
 * Este archivo se distribuye bajo los términos de la licencia GNU. Para más
 * información, consultar el archivo
 * LICENCIA.pdf o http://es.tldp.org/Otros/gples/gples.html
 *
 */

#include <stdio.h>
#include "matriz.h"
#include "dni.h"

main()
{
  /* Dos matrices A y B */
       Matriz A, B;
       DNI un_dni;
       int i,j;
       float tmp;
      pon_datos_dni("Juan", "Rodríguez", "Vázquez", "V", "soltero",
      "Cervantes", "47019", "Valladolid","4445356-J", &un_dni);
```

```
 /* En una meteremos DNIs y en la otra números reales */
 MatrizCrea(&A, 3, 3, sizeof(DNI));
 MatrizCrea(&B, 3, 3, sizeof(float));
 for(i=0;i<3;i++)
 {
       for(j=0;j<3;j++)
       {
             tmp = (float) i * j;
             MatrizCambia(&A,i,j,&un_dni);
             MatrizCambia(&B,i,j,&tmp);
       }
 }
 for(i=0;i<3;i++)
 {
       for(j=0;j<3;j++)
       {
            MatrizObten(&A,i,j,&un_dni);
            imprime_dni(un_dni);
       }
       printf("\n");
 }
 for(i=0;i<3;i++)
 {
       for(j=0;j<3;j++)
       {
            MatrizObten(&B,i,j,&tmp);
            printf("%2.2f ", tmp);
       }
       printf("\n");
 }
}
```

Debido a la extensión de este artículo, dejaremos para el siguiente el cómo crear un Makefile para este proyecto. Por ahora podremos compilar estos archivos de la siguiente manera:

```
luis@leonov:~/hxc/articulo7/ejemplos/matriz$ cc ppal.c matriz.c dni.c -I./ -o matriz

luis@leonov:~/hxc/articulo7/ejemplos/matriz$ ./matriz
```

```
.....

.....

Calle: Cervantes, 47019, Valladolid

S: V   E.C.: soltero


DNI: 4445356-J

Nombre: Juan

Apellidos: Rodríguez Vázquez

Calle: Cervantes, 47019, Valladolid

S: V   E.C.: soltero


0.00 0.00 0.00

0.00 1.00 2.00

0.00 2.00 4.00

luis@leonov:~/hxc/articulo7/ejemplos/matriz$
```

Bibliografía

man printf; man scanf, Varios.

UNiX Programación Avanzada, Fco. Manuel Márquez, Editado por Ra-Ma, ISBN: 84-7897-239-0.

Slackware Linux Unleashed, Bao Ha y Tina Nguyen, Editado por Pearson Education, ISBN: 0-672-31768-0.

Advanced Linux Programming, Mark Mitchel, Jeffrey Oldham, y Alex Samuel, Editado por New Riders, ISBN: 0-7357-1043-0.

MANIPULACION DE DOCUMENTOS XML EL DOM SEGUNDA PARTE INTERFAZ XMLDOMNODE (I)

POR JOAQUIM ROCA VERGES

Solo aquellos que han seguido el curso de XML hasta aquí podrán seguir avanzando. Si, lo sabemos, XML es complejo hasta que uno le dedica las horas necesarias... después la recompensa llega por si sola :)

En el pasado número veíamos la teoría del DOM.

Recordemos que el DOM es una manera de ver el archivo xml, para que nos sea fácil navegar por el documento, editarlo, copiarlo etc.

Como conceptos básicos teníamos:

- **DOM** = modelo de objetos del documento xml. Todo el documento XML, todo lo que contiene el documento xml.

- **Interfaz =** conjunto de métodos y propiedades, agrupados según un criterio. Agrupados según una manera de ver las cosas.

- **Objeto** = todos los componentes (elementos, notaciones, atributos, texto...) del xml se consideran objetos. Cada uno de los objetos pertenece a una interfaz determinada.

- **Nodo =** todo elemento, texto... del xml se considera un nodo. Todo objeto del xml es un nodo.

- **Interfaz de nodos =** un conjunto de métodos (funciones) y propiedades para los objetos (elementos, comentarios, atributos, entidades.. del documento XML) DOM.

Como podéis ver, es algo abstracto y difícil de imaginar, hasta que no tenéis un ejemplo delante. No os preocupéis mucho si de entrada no lo entendéis bien. Lo importante es que comprendáis bien los ejemplos, y cuando tengáis claro los ejemplos, seguramente entenderéis todos estos conceptos abstractos que os he enumerado.

Vimos la interfaz DOMDocument y sus principales propiedades y métodos como podrían ser **load , save y** mencionamos **los métodos create.** En este número continuaremos explicando la interfaz XMLDOMNode, en la que veremos , entre otras cosas estos métodos create, que si bien pertenecían a DOMDocument se comprenden mejor desde la interfaz XMLDOMNode **.**

Antes de comenzar, unas palabras de aliento, con una nueva experiencia XML :

He estado trabajando con un gestor de contenidos Web, que se llama teamsite de la casa interwoven. Sirve para mantener una website de manera cómoda con una serie de plantillas que ellos llaman templates. Estas plantillas son útiles por ejemplo para gestionar las descripciones de los productos. Supongamos una multinacional que antes y después de la descripción del producto quiere poner un texto aclaratorio. Este texto es común a algunos productos. Pues con teamsite se seleccionan las plantillas y se les indica en que productos va a salir la descripción.

La estructura de las plantillas está en... XML, que se valida contra un DTD. Se recorre el DOM XML con PERL y Visual Basic.

INTERFAZ DOMNode

Podemos considerar esta interfaz como el centro del DOM: el nodo en persona.

El nodo es la unidad fundamental del DOM. Hay varios tipos de objetos nodo: algunos pueden tener nodos hijos, mientras que otros son nodos finales (se les denomina **leaf nodes = nodos hoja)**, esto es, nodos que no tienen ni pueden tener nodos hijos.

La interfaz DOMNode define un conjunto de métodos y atributos genéricos, pero no todos ellos son aplicables a todos los tipos de nodos. Por ejemplo: existe un atributo que se llama **childnodes** que devuelve un **nodelist** que contiene los hijos de un nodo determinado. Tú preguntas por los nodos hijo (childnodes) de un nodo determinado y te devuelve una lista con todos los nodos (nodelist) hijo de ese nodo determinado.

Sin embargo, el nodo de tipo TEXT no puede tener nodos hijo, con lo que no se puede aplicarle el atributo childnodes.

Tabla de tipos de nodos

Tipo de nodo	Nodos hijos permitidos	Nombre de la constante	Valor
Elemento	Elemento, Texto, Comentario, Instrucción de procesamiento, Sección CDATA, Referencia a entidad.	NODE_ELEMENT	1
Atributo	Texto, Referencia a Entidad	NODE_ATTRIBUTE	2
Texto	Sin hijos	NODE_TEXT	3
Sección CDATA	Sin Hijos	NODE_CDATA_SECTION	4
Referencia a Entidad	Elemento, Instrucción de procesamiento, Comentario, Texto, Sección CDATA, Referencia a Entidad	NODE_ENTITY_REFERENCE	5
Entidad	Elemento, Instrucción de procesamiento, Comentario, Texto, Sección CDATA, Referencia a Entidad	NODE_ENTITY	6
Instrucción de procesamie nto	Sin hijos	NODE_PROCESSING_INST RUCTION	7
Comentario	Sin Hijos	NODE_COMMENT	8
Documento	Elemento (uno como máximo), Instrucción de procesamiento, Comentario, Tipo de documento	NODE_DOCUMENT	9
Tipo de documento	Sin Hijos	NODE_DOCUMENT_TYPE	10
Fragmento	Elemento, Instrucción de procesamiento, Comentario, Texto, Sección CDATA, Referencia a Entidad	NODE_DOCUMENT_FRAGM ENT	11
Notación	Sin hijos	NODE_NOTATION	12

TABLA DE ATRIBUTOS DE LA INTERFAZ DOMNODE

Vamos a ver una tabla mas, la de atributos genéricos para la interfaz DOMNODE.
Los valores de los atributos **nodename** y **nodevalue** dependen del tipo de nodo a que nos estemos refiriendo. Por ejemplo el valor de un nodo elemento es siempre **null** mientras que el valor para un nodo atributo es el texto del atributo.

Si preguntamos por el valor de un elemento con la propiedad **nodevalue** no conseguiremos el texto que contiene el elemento, sin embargo, el elemento seguramente que contiene nodos hijos de tipo NODE_TEXT o NODE_CDATA_SECTION y es el **nodevalue** de estos nodos (de los nodos texto o cdata) el que nos devolverá el texto.

Nombre del atributo	Descripción
NodeName	Nombre del nodo, dependiendo de su tipo. Es decir que varia según el tipo de nodo. Por ejemplo , para un nodo atributo contiene el nombre del atributo, pero para un nodo text, contiene siempre el literal "#text".
NodeValue	Valor del nodo, tal como se ha explicado antes, dependiendo del tipo de nodo que sea.
NodeType	Tipo de nodo. Código del tipo de nodo
ParentNode	Padre del nodo. Todos los nodos a excepción del nodo documento, el nodo fragmento y el nodo atributo pueden tener un nodo padre. Sin embargo, si un nodo ha sido creado pero no ha sido añadido todavía al árbol, o ha sido borrado del árbol, tiene como valor un valor nulo.
ChildNodes	Un NodeList (una lista de nodos) que contiene todos los hijos de este nodo. Si no tiene hijos, es un NodeList sin nodos. El contenido del tipo NodeList devuelto por este atributo está "vivo", en el sentido de que los cambios que se hagan en el nodo que devolvió el NodeList, se reflejaran inmediatamente en este NodeList devuelto. Por ejemplo, un nodo sin hijos devuelve un nodelist vacío, pero si ha este nodo le añado un nodo hijo, el nodelists automáticamente se actualiza y contiene el nuevo nodo hijo. Esto es válido para todos los Nodelist devueltos, incluidos los devueltos por el método **getElementsByTagName.**
FirstChild	El primer nodo hijo de este nodo. Si no tiene hijos devuelve un valor de tipo null.
LastChild	El último nodo hijo de este nodo. Si no tiene hijos devuelve un valor de tipo null.
PreviousSibling	El nodo inmediatamente anterior a este nodo. Si no tiene un nodo anterior, devuelve un tipo nulo.
NextSibling	El nodo inmediatamente posterior a este nodo. Si no tiene un nodo posterior, devuelve un tipo nulo
Attributes	Devuelve un tipo **NamedNodeMap** que contiene los atributos de este nodo, si no tuviera atributos devuelve un nulo
OwnerDocument	El documento asociado a este nodo. Cuando el nodo es el documento mismo, devuelve un nulo.

Vamos a empezar a practicar con esta interfaz.
Recordamos que teníamos una carpeta en la raíz de C, en C:\xmlDOM. Y dentro de esta carpeta, un par de dtd, un par de xml y el proyecto visual basic.
Recordamos el dtd pedidos.dtd, y le añadimos la posibilidad de que haya mas de un producto por orden de compra, por lo tanto, pondremos un signo más +, al lado de PRODUCTO, tal como sigue:

```
<!ELEMENT PEDIDOS (ORDEN_DE_COMPRA+)>
        <!ELEMENT ORDEN_DE_COMPRA (CLIENTE, PRODUCTO+)>
        <!ELEMENT CLIENTE (NUMERO_DE_CUENTA,NOMBRE_COMPLETO)>
        <!ELEMENT PRODUCTO (#PCDATA)>
        <!ELEMENT NUMERO_DE_CUENTA ( #PCDATA)>
        <!ELEMENT NOMBRE_COMPLETO ( NOMBRE, APELLIDO1,APELLIDO2) >
        <!ELEMENT NOMBRE (#PCDATA)>
        <!ELEMENT APELLIDO1 (#PCDATA)>
        <!ELEMENT APELLIDO2 (#PCDATA)>
```

Recordamos el xml pedidos.xml y le añadimos un producto:

```
<?xml version="1.0" standalone="no" ?>
<!DOCTYPE PEDIDOS  SYSTEM  "pedidos.dtd">
<PEDIDOS>
        <ORDEN_DE_COMPRA>
        <CLIENTE>
                <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
                <NOMBRE_COMPLETO>
                        <NOMBRE>Sam</NOMBRE>
                        <APELLIDO1>Bass</APELLIDO1>
                        <APELLIDO2></APELLIDO2>
                </NOMBRE_COMPLETO>
        </CLIENTE>
        <PRODUCTO>LIBRO</PRODUCTO>
        </ORDEN_DE_COMPRA>
        <ORDEN_DE_COMPRA>
        <CLIENTE>
                <NUMERO_DE_CUENTA>987654321</NUMERO_DE_CUENTA>
                <NOMBRE_COMPLETO>
                        <NOMBRE>Jesse</NOMBRE>
                        <APELLIDO1>James</APELLIDO1>
                        <APELLIDO2></APELLIDO2>
                </NOMBRE_COMPLETO>
        </CLIENTE>
        <PRODUCTO>DISCO DE VINILO</PRODUCTO>
        <PRODUCTO>SUPERSNAZZ -  FLAMIN' GROOVIES</PRODUCTO>

        </ORDEN_DE_COMPRA>
</PEDIDOS>
```

MÉTODOS DE LA INTERFAZ DOMNODE

Vamos a analizar en profundidad los métodos genéricos de esta interfaz.

MÉTODO APPENDCHILD

Sintaxis: nodo.appendchild nuevo_nodo

Este método añade el nodo **nuevo_nodo** (donde **nuevo_nodo** es el nodo que añade) al final de la lista de nodos hijos de **nodo**.

Esto es, si añadimos un nuevo nodo <PRODUCTO> a <ORDEN_DE_COMPRA> , lo colocaria justo debajo de :

```
<PRODUCTO>SUPERSNAZZ -  FLAMIN' GROOVIES</PRODUCTO>
```

Recordemos que todo es un nodo, así si le decimos que añada un nodo PRODUCTO a ORDEN_DE_COMPRA y no le decimos nada mas, lo añadirá como un nodo vacío (repasad los elementos vacíos en el nº10 de la revista), como un elemento sin contenido(<PRODUCTO/> es decir que el nuevo xml, quedará de la siguiente manera:

```
<PRODUCTO>SUPERSNAZZ -  FLAMIN' GROOVIES</PRODUCTO>
<PRODUCTO/>
```

Para que no nos quede como un elemento vacío tendremos que ponerle contenido al nuevo nodo, y una manera fácil de hacer es escribiendo la propiedad **text:**

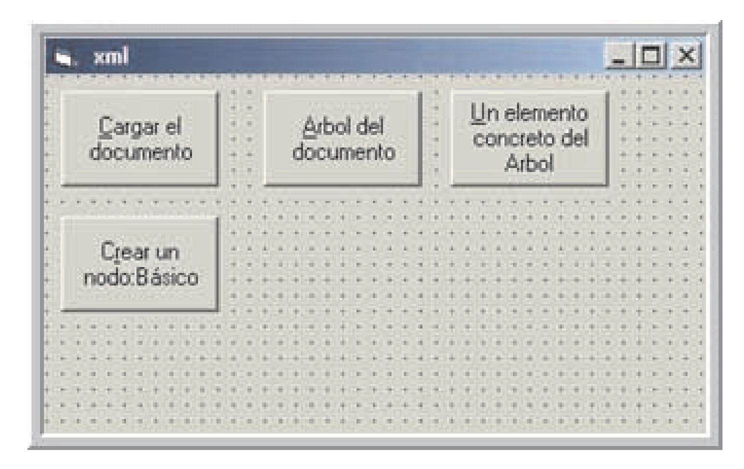
Nuevonodo.text = "TEXTO DEL NUEVO NODO". Y ahora si, cuando hagamos el appendchild, el nuevo xml nos quedará:

```
<PRODUCTO>SUPERSNAZZ -  FLAMIN' GROOVIES</PRODUCTO>
<PRODUCTO> TEXTO DEL NUEVO NODO </PRODUCTO>
```

Decíamos que el método **appendchild,** añadía un nuevo nodo, si el nodo fuera un nodo **documentFragment**, el contenido entero del fragmento de documento se añadiría al nodo padre que estuviéramos manipulando, o que tuviéramos seleccionado.

Abrid el Visual Basic y seleccionad el proyecto que empezamos el número anterior.
Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdCrearNodoBasico
- **caption** = C&rear un nodo: Básico



Codificamos el evento click del botón:

```
'**************Escribid**************************
Dim xmlDocument As New DOMDocument
Dim nodoPadre As IXMLDOMNode
Dim nodoNuevo As IXMLDOMNode
Dim nodoElemento As IXMLDOMElement
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
'añadimos una referencia a un nodo, en este caso el nodo raiz
Set nodoPadre = xmlDocument.documentElement
'creamos un nodo elemento: los métodos create los enumeramos en el anterior número,
'ahora los vemos en acción
Set nodoElemento = xmlDocument.createElement("PEPE")
'añadimos texto al nuevo elemento, de lo contrario nos saldría un elemento vacío
'haced la prueba, comentar la siguiente línea, y veréis que en lugar de un elemento con
'contenido os sale un elemento vacío <PEPE/>
nodoElemento.text="Suerte que ahora mismo no validamos contra un DTD"
'pegamos el nodo al documento, y lo colocamos en la variable de tipo Node nuevoNodo
Set nuevoNodo = nodoPadre.appendChild(nodoElemento)
'guardamos las modificaciones
xmlDocument.save ("C:\xmlDom\pedidos.xml")
MsgBox xmlDocument.xml
'***********************************************
```

Vamos a analizarlo un poco.
Fijaros que ha añadido el nuevo elemento justo debajo del segundo nodo ORDEN_DE_COMPRA:

```
        </ORDEN_DE_COMPRA>
        <PEPE>Suerte que ahora mismo no validamos contra un DTD</PEPE>
</PEDIDOS>
```

Os podéis preguntar: ¿Y porque justamente ahí?. Pues justamente ahí, porque es ahí donde le hemos dicho que lo añadiera.

Veamos de nuevo el código:

```
'añadimos una referencia a un nodo, en este caso el nodo raiz
Set nodoPadre = xmlDocument.documentElement
```

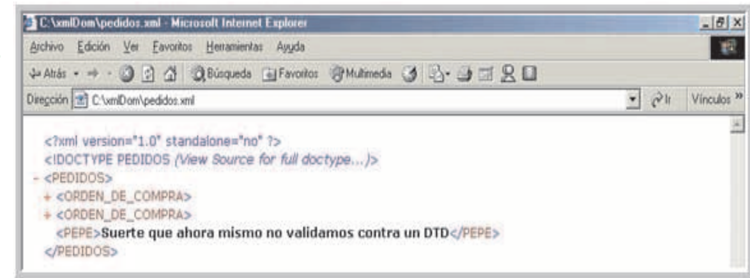
Y unas líneas más abajo pone:

```
'pegamos el nodo al documento, y lo colocamos en la variable de tipo Node nuevoNodo
Set nuevoNodo = nodoPadre.appendChild(nodoElemento)
```

Es decir que estamos pegando el nuevo nodo al nodoPadre, y ¿Cuál es el nodoPadre? Pues **en este caso** el nodo raiz, o sea el nodo PEDIDOS.
Si nos hubiéramos situado en otro nodo, si le hubiéramos dicho que el nodoPadre era otro, lo hubiera pegado en el nodo que le hubiéramos dicho. ¿Y como le decimos que el nodo al que vamos a pegar el nuevo nodo es por ejemplo PRODUCTOS? Seguid leyendo y lo averiguareis.

Fijaros en el texto que le añadido a nodoElemento: "Suerte que ahora mismo no validamos contra un DTD". Este nuevo elemento PEPE no cumple con las reglas del DTD pedidos.dtd, por tanto ha convertido el archivo pedidos.xml en un archivo **no válido.**

No obstante, si abrís el archivo en Internet Explorer veréis en vuestras pantallas lo siguiente:



y os podéis preguntar como es posible que explorer de por bueno un archivo xml no válido. ¿Qué hemos hecho mal?

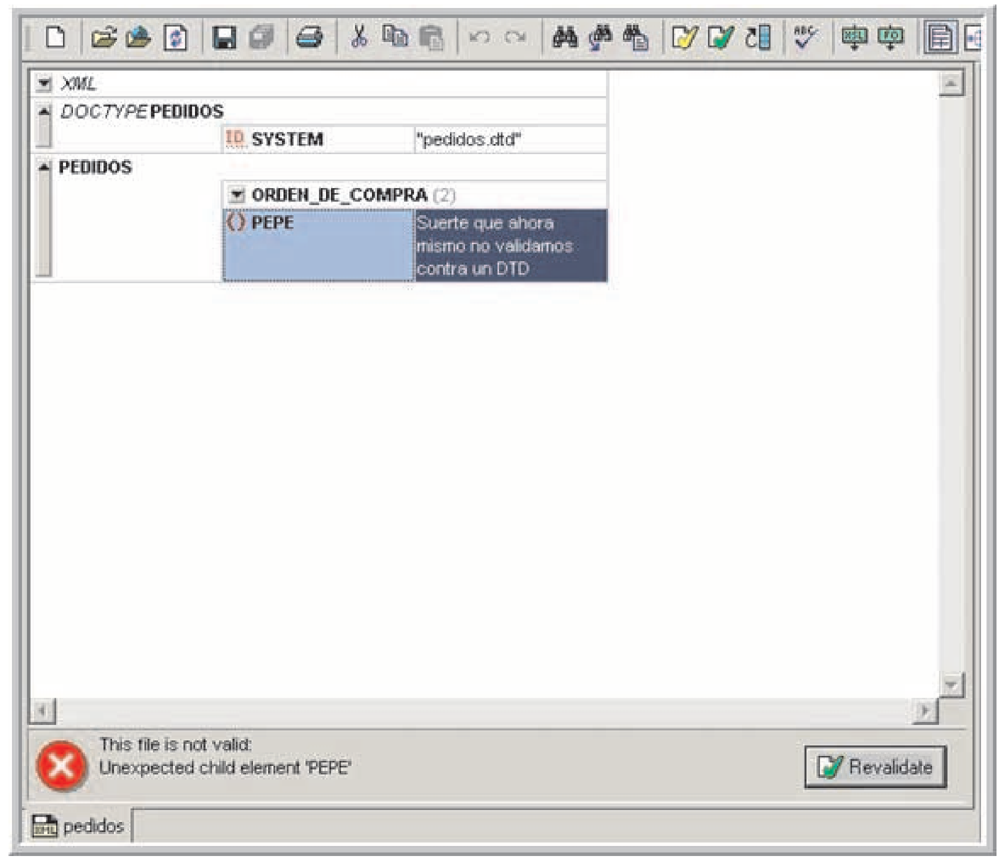
Pues lo que hemos hecho mal es confiar en Microsoft ya que a pesar de que el xml se válida contra el DTD, y el DTD nos dice que no es un xml válido, lo muestra en pantalla como si fuera un xml válido. De lo que deducimos que la validación de archivos xml contra DTDs
en el explorer de Microsoft no está bien resuelta.

Y ya que ha salido el tema, si me permitís me gustaría daros mi opinión sobre el debate Microsoft-Si, Microsoft-No.
Pienso por un lado que la seguridad de microsoft en sistemas operativos es francamente mala, que sus sistemas operativos dejan bastante que desear, que sus productos salen bastante tiempo antes de lo que debieran al mercado , que el hecho de ir sacando un service pack tras otro de sus productos aunque ya nos hemos acostumbrado, si lo miramos fríamente y lo analizamos un poco es algo vergonzoso tanto para la gente de Microsoft como para nosotros.
Pero por otro lado pienso que debemos agradecerle que haya hecho el PC tan popular, que haya convertido al ordenador algo tan común en los hogares como la nevera o la lavadora. Y al ser tan popular y generar tantos beneficios haya forzado al abaratamiento de las piezas y a la rápida evolución de velocidades, capacidades de disco, y haya favorecido la aparición de nuevos lenguajes, técnicas y arquitecturas.

Nos vamos a asegurar de que no es un xml válido.
Hay una herramienta francamente buena para construir archivos xml y DTDs, se llama XML SPY, y podéis bajárosla de http://www.altova.com/download.html.

Con esta herramienta, podéis construir xml, saber si están bien formados y si no son standalone validarlos contra un dtd.

Veamos pues con una imagen el resultado de la validación del xml con xml spy:



Fijaros en el mensaje de error que hay al lado de la X :

"This file is not valid: Unexpected child element "PEPE".

Me gustaría que quedara algo muy claro de esta primera aproximación:

1- Referenciar siempre al nodopadre donde vamos a insertar un hijo, es decir tenemos que apuntar al lugar donde vamos a insertar el nuevo nodo. Si queremos situar el nuevo nodo

en un lugar determinado deberemos decirle al programa el nodo donde vamos a insertarlo

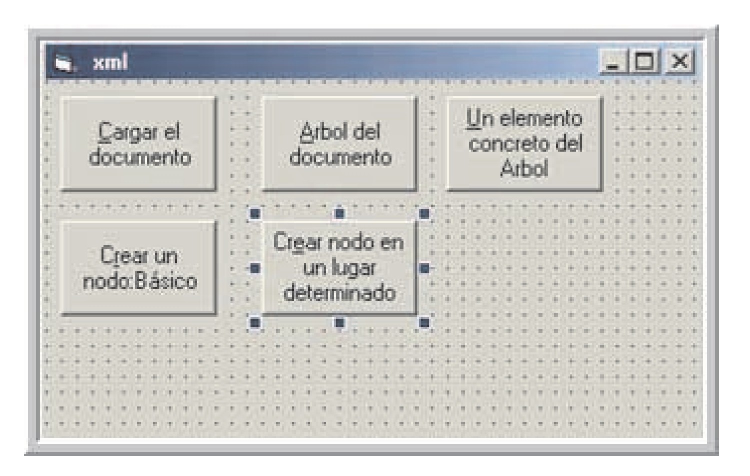
2- Crear el tipo de nodo que vamos a insertar y asignarle texto, atributos... lo que haga falta

3- Enganchar el nodo. Y lo engancharemos exactamente en el lugar que le hayamos indicado en el punto 1

Vamos a ver un ejemplo de cómo **insertar un nodo en un lugar determinado** del documento, en el lugar que nosotros queramos.

Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdNodoDeterminado
- **caption** = Cr&ear nodo en un lugar determinado



Codificad el evento click del botón

```
'*************Escribid****************
Dim xmlDocument As DOMDocument
Dim nuevoElemento As IXMLDOMElement
Dim nuevoNodo As IXMLDOMNode
Dim nodoBase As IXMLDOMNode
Dim listadenodos As IXMLDOMNodeList
Dim xmlString As String
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
'creamos el nodo y le asignamos el contenido. Fijaros que hemos
'cambiado el método de createelement a createnode, método al
'que le decimos que el nuevo nodo va a ser de tipo elemento, y el
'nombre que va a tener el nodo. El tercer parámetro lo dejamos vacío
Set nuevoNodo = xmlDocument.createNode(NODE_ELEMENT, "PRODUCTO", "")
nuevoNodo.Text = "AFTERMATH -- ROLLING STONES"
'le decimos cual va a ser el nodo padre, cogiendo ORDEN_DE_COMPRA como
'nodo padre, se insertará pues dentro del nodo ORDEN_DE_COMPRA,
Set listadenodos = xmlDocument.getElementsByTagName("ORDEN_DE_COMPRA")
'INSERTAMOS DESPUES DEL PRIMERO –DESPUES DEL PRIMER
'ORDEN_DE_COMPRA
'Set nodoBase = listadenodos.Item(0)
'INSERTAMOS DESPUES DEL ÚLTIMO—DESPUES DEL ULTIMO
'ORDEN_DE_COMPRA
Set nodoBase = listadenodos.Item(listadenodos.length - 1)
nodoBase.appendChild nuevoNodo
MsgBox xmlDocument.xml
'*************************************
```

Ahora si que lo hemos insertado de manera correcta:

```
    <PRODUCTO>DISCO DE VINILO</PRODUCTO>
    <PRODUCTO>SUPERSNAZZ - FLAMIN'GROOVIES</PRODUCTO>
    <PRODUCTO>AFTERMATH -- ROLLING STONES</PRODUCTO>
</ORDEN_DE_COMPRA>
```

Esto ocurre en el caso de que el nodo base sea el último, y esto se lo decimos asignando al nodoBase el ultimo elemento de listadenodos (recordemos que lista de nodos tiene todos los elementos ORDEN_DE_COMPRA y ORDEN_DE_COMPRA es el nodo que estamos haciendo servir de nodo base o nodo padre)

Set nodoBase = listadenodos.Item(listadenodos.length - 1)

listadenodos.length = tiene el numero de nodos ORDEN_DE_COMPRA que son **DOS**, como las matrices se numeran partiendo de 0, la matriz de nodos listadenodos tendrá dos elementos: 0 y 1.

El elemento UNO, el nodo UNO corresponderá a listadenodos.Item(**0**)

El elemento DOS, el nodo DOS a listadenodos.Item(**1**)
Listadenodos.length = 2

Set nodoBase = listadenodos.Item(**2** - 1) o sea listadenodos.item(1).

En el caso que le indicaramos que el nodobase fuera el primero:

Set nodoBase = listadenodos.Item(0), el resultado sería:

<PRODUCTO>LIBRO</PRODUCTO>
<PRODUCTO>AFTERMATH – ROLLING STONES</PRODUCTO>
</ORDEN_DE_COMPRA>

Analicemos un poco:
Hemos cogido a ORDEN_DE_COMPRA y le hemos añadido un hijo con el método **appendchild**. Los hijos siempre se añaden al final, por lo tanto solo podemos colocar el nuevo nodo después del nodo <PRODUCTO>LIBRO</PRODUCTO> si cogemos como nodo padre el primer ORDEN_DE_COMPRA o después del nodo <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO> si cogemos como nodo padre el segundo y último nodo ORDEN_DE_COMPRA.

¿cómo hacemos para insertar un nuevo nodo producto antes de <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO> o **entre** <PRODUCTO>DISCO DE VINILO</PRODUCTO> **y** <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO>?

Pues con el siguiente método : **insertBefore**

MÉTODO INSERTBEFORE

Sintaxis: insertBefore (node nuevoHijo, node nodoReferencia)

Este método añade un nodo hijo a un nodo padre, en una posición especifica de la lista de nodos hijos. Por ejemplo , si cogemos el segundo nodo ORDEN_DE_COMPRA como nodo padre, su lista de nodos hijos PRODUCTO será:

<PRODUCTO>DISCO DE VINILO</PRODUCTO> y <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO> y podremos insertar el nodo donde queramos, antes o después <PRODUCTO>DISCO DE VINILO</PRODUCTO>; o Antes o después de <PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO>

Vamos a realizar un ejemplo con el visual basic:

Vamos a insertar un nodo entre

<PRODUCTO>DISCO DE VINILO</PRODUCTO>

y

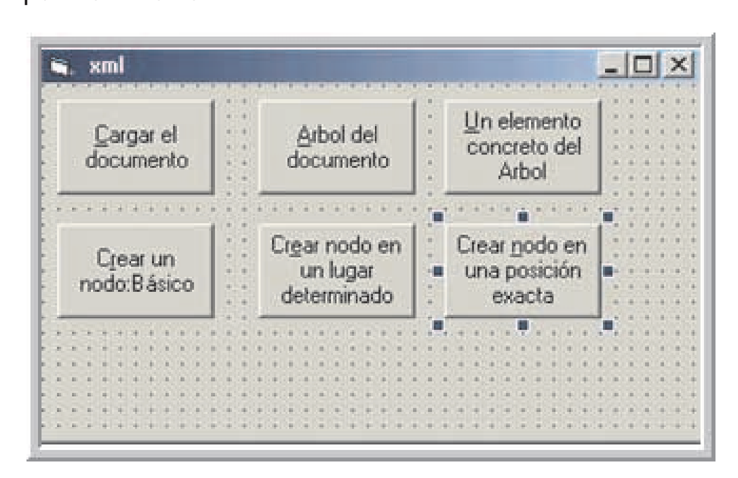
<PRODUCTO>SUPERSNAZZ - FLAMIN' GROOVIES</PRODUCTO>

y después insertar un nodo justo antes de

<PRODUCTO>LIBRO</PRODUCTO>

Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdCrearExacto
- **caption** = Crear &nodo en una posición exacta



Codificad el evento del botón:

```
Private Sub cmdCrearExacto_Click()
Dim xmlDocument As DOMDocument
Dim nuevoNodo As IXMLDOMNode
Dim lstNodosPadre As IXMLDOMNodeList
Dim lstNodosReferencia As IXMLDOMNodeList
Dim nodoReferencia As IXMLDOMNode
Dim nodoPadre As IXMLDOMNode
Dim xmlString As String
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")
```

```
'nodo que vamos a insertar, lo creamos y queda "suelto" hasta
'que lo enganchemos a otro nodo

Set nuevoNodo = xmlDocument.createNode(NODE_ELEMENT, "PRODUCTO", "")
nuevoNodo.Text = "AFTERMATH -- ROLLING STONES"

'Lista de nodos del padre donde lo vamos a enganchar:
'Si lo insertamos justo encima de <PRODUCTO>LIBRO</PRODUCTO>
'tenemos que darle al método insertBefore el padre de PRODUCTO,
'que es ORDEN_DE_COMPRA.

Set lstNodosPadre = xmlDocument.getElementsByTagName("ORDEN_DE_COMPRA")

'Una vez tenemos al nodo padre, insertbefore debe saber con
'respecto a que nodo hijo de ORDEN_DE_COMPRA vamos a
'insertar, necesitaremos pues saber el nodo "hermano" (sibling)
'con respecto al cual vamos a insertar. Técnicamente se llama el
'nodo de referencia.Fijaros que he escrito:
'getElementsByTagName("ORDEN_DE_COMPRA/PRODUCTO").
'Para que veáis como se referencia un nodo conjuntamente con
'su padre también sería válido hacerlo como  hasta ahora:
'getElementsByTagName("PRODUCTO").
'¡Probadlo!, tocad el código, hacer vuestras pruebas y
'experimentos!!

Set lstNodosReferencia = xmlDocument.getElementsByTagName("ORDEN_DE_COMPRA/PRODUCTO")

'teníamos la lista de nodos padre, pero no el nodo padre, como
'queremos insertar justo encima de
'<PRODUCTO>LIBRO</PRODUCTO>, y este nodo
'está en el primer ORDEN_DE_COMPRA, es decir, en el primer
'nodo padre le decimos que el nodoPadre es el primer item (también
'se le llama así cuando hay un array de objetos) de la lista de
'nodos, el primer nodo de la lista de nodos padre.

Set nodoPadre = lstNodosPadre.Item(0)

'Si quisiéramos insertar entre <PRODUCTO>DISCO DE
'VINILO</PRODUCTO> y <PRODUCTO>SUPERSNAZZ -
'FLAMIN' GROOVIES</PRODUCTO>
'referenciaríamos el segundo nodoPadre:
'Set nodoPadre = lstNodosPadre.Item(1)
'¡Probadlo!

'Hasta ahora le hemos preparado para hacer un insertbefore(inserta
'antes) dentro del primer nodo padre ORDEN_DE_COMPRA.
'Imaginad que ya estamos dentro de el primer nodo
'ORDEN_DE_COMPRA. ahora nos falta decirle al método cual
'va a ser el nodo hermano, donde va a estar el nodo hermano
'PRODUCTO dentro de ORDEN_DE_COMPRA con respecto al
'cual vamos a hacer un insertbefore(inserta antes).
'Queríamos insertar antes de LIBRO, pues referenciados
'<PRODUCTO>LIBRO</PRODUCTO>

Set nodoReferencia = lstNodosReferencia.Item(0)

'si quisiéramos insertar entre <PRODUCTO>DISCO DE
'VINILO</PRODUCTO> y <PRODUCTO>SUPERSNAZZ -
'FLAMIN' GROOVIES</PRODUCTO> deberíamos referenciar
'<PRODUCTO>SUPERSNAZZ - FLAMIN'
'GROOVIES</PRODUCTO> para que insertbefore, insertara
'justo antes de ese elemento y ese elemento es el tercero de la lista
'de nodos de referencia, que son los nodos PRODUCTO, entonces
'escribiríamos Set nodoReferencia = lstNodosReferencia.Item(2)
' ¡Probadlo! Y acordaros que para que funcione tenéis que
referenciar el segundo ' nodoPadre: 'Set nodoPadre =
lstNodosPadre.Item(1)

'Uno puede preguntarse, ¿Siempre es necesario crear una variable
'del tipo NodeList para referenciar un nodo determinado?.
'La respuesta es : ¡No!, hay otras maneras. Por ejemplo para
'asignar el nodo de referencia podríamos escribir:
'Set nodoReferencia = xmlDocument.document
'Element.childNodes(0).childNodes(0)
'fijaros: xmlDocument, que es el elemento raíz, .childnodes(0) que
'es el primer elemento ORDEN_DE_COMPRA y dentro de el
'primer ORDEN_DE_COMPRA, el primer PRODUCTO!!
'para asignar el nodo padre, escribiríamos:
'Set nodoPadre = xmlDocument.documentElement.childNodes(0)
' ¡Jugad con los atributos que ya conocéis!, investigad , probad
'métodos de la interfaz DOMDocument!!

'Finalmente ejecutamos el insertBerfore, le decimos que el
'nodoPadre inserte el nuevoNodo antes del nodoReferencia, y lo
'asignamos a la variable nuevoNodo, para que apunte al nodo
'nuevo que hemos insertado dentro del árbol ya que hasta ahora
' apuntaba al nuevo nodo pero este nodo estaba "suelto". Después
'de insertBefore seguirá apuntando al nuevoNodo, que ya no estará
'suelto sino dentro del árbol de nodos.

Set nuevoNodo = nodoPadre.insertBefore(nuevoNodo, nodoReferencia)

MsgBox xmlDocument.xml

End Sub
```

MÉTODOS REPLACECHILD, REMOVECHILD Y CLONENODE

Sintaxis: replaceChild (node nuevoHijo, node nodoAReemplazar)

Si en lugar de añadir un nodo, lo que queremos hacer es reemplazarlo, este es el método que debemos utilizar. Este método reemplaza **nodoAReemplazar** por **nuevoHijo** y devuelve una referencia al nodo reemplazado **Sintaxis-2 removechild (node nodoBorrar)**

Tal como su nombre indica **remove (borrar) child (hijo)**, este método se utiliza para borrar el nodo que se pasa como parámetro (nodoBorrar)

Sintaxis-3 cloneNode (boolean profundidad)

En caso de que haya algún nodo en particular que queráis clonar de un sitio del DOM a otro, podéis clonarlo. El método **cloneNode** devuelve una referencia al nodo duplicado (clonado). El parámetro profundidad (**deep)** puede tomar los valores true o false. Pondremos true cuando queramos que nos copie el nodo y todos sus nodos hijos. Por ejemplo si clonáramos el primer nodo ORDEN_DE_COMPRA y le pasásemos un true, nos copiaría enterito el fragmento xml:

```
<ORDEN_DE_COMPRA>
<CLIENTE>
<NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
        <NOMBRE_COMPLETO>
                <NOMBRE>Sam</NOMBRE>
                <APELLIDO1>Bass</APELLIDO1>
                <APELLIDO2></APELLIDO2>
        </NOMBRE_COMPLETO>
</CLIENTE>
<PRODUCTO>LIBRO</PRODUCTO>
</ORDEN_DE_COMPRA>
```

y si le pasáramos un false solo nos duplicaría el nodo, sin sus nodos hijos , esto es:

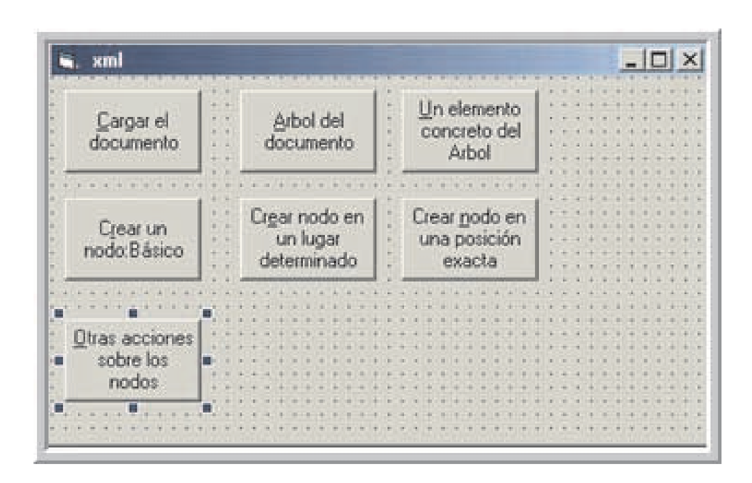
```
<ORDEN_DE_COMPRA>
</ORDEN_DE_COMPRA>.
```

Hay que tener en cuenta que el nodo clonado, no está enganchado a ninguna parte del documento xml, esta suelto, y tendremos que hacer un **appendchild** o un **insertbefore** para pegarlo al documento.

Vemos estos tres métodos con un ejemplo:

Colocad un botón en el formulario, id a las propiedades y poner:

- **name** = cmdOtrasAccionesNodos
- **caption** = &Otras acciones sobre los nodos



Codificad el evento click del botón:

```
Private Sub cmdOtrasAccionesNodos_Click()

Dim xmlDocument As DOMDocument
Dim nodoPadre As IXMLDOMNode
Dim nuevoNodo As IXMLDOMNode
Dim nodoReemplazar As IXMLDOMNode
Dim nodoClonado As IXMLDOMNode
Dim xmlString As String
Set xmlDocument = New DOMDocument
xmlDocument.async = False
xmlDocument.validateOnParse = True
xmlDocument.resolveExternals = True
xmlDocument.Load ("C:\xmlDom\pedidos.xml")

'asignamos como nodoPadre, el primer nodo
'ORDEN_DE_COMPRA, lo podéis comprobar con el mensaje
'msgbox

Set nodoPadre = xmlDocument.documentElement.childNodes(0)
MsgBox nodoPadre.xml
```

```
'el nuevo nodo será un nodo <PRODUCTO>LIBRO
'ELECTRONICO</PRODUCTO> ya que lo que vamos a hacer
'en primer lugar es reemplazar:
'<PRODUCTO>LIBRO</PRODUCTO> por
'<PRODUCTO>LIBRO ELECTRONICO</PRODUCTO>

Set nuevoNodo = xmlDocument.createNode(NODE_ELEMENT, "PRODUCTO", "")
nuevoNodo.Text = "LIBRO ELECTRONICO"

'asignamos a nodoReemplazar el nodo <PRODUCTO>LIBRO </PRODUCTO>
'lo comprobamos con un mensaje

Set nodoReemplazar = xmlDocument.documentElement.childNodes(0).childNodes(1)
MsgBox nodoReemplazar.xml

'reemplazamos nodoReemplazar por nuevoNodo y lo comprobamos con un mensaje

Set nuevoNodo = nodoPadre.replaceChild(nuevoNodo, nodoReemplazar)
MsgBox nodoPadre.xml

'vamos a clonar todo el nodo padre, esto es  el nodo y su contenido,
'el nodo y sus hijos:
'   <ORDEN_DE_COMPRA>
'   <CLIENTE>
'    <NUMERO_DE_CUENTA>12345678</NUMERO_DE_CUENTA>
'      <NOMBRE_COMPLETO>
'         <NOMBRE>Sam</NOMBRE>
'         <APELLIDO1>Bass</APELLIDO1>
'         <APELLIDO2></APELLIDO2>
'      </NOMBRE_COMPLETO>
'   </CLIENTE>
'   <PRODUCTO>LIBRO</PRODUCTO>
'   </ORDEN_DE_COMPRA>

'le decimos que lo clone todo, poniendo el parámetro profundidad (deep) a True

Set nodoClonado = nodoPadre.cloneNode(True)
Msgbox nodoClonado.xml

'Ya que una vez clonado el nodo, lo queremos enganchar al
'documento xml , vamos a cambiar el nodo padre, cambiamos la
'referencia del nodo padre, ahora pasa a ser de
'ORDEN_DE_COMPRA a PEDIDOS, ya que lo que vamos hacer
'es enganchar un nuevo nodo ORDEN_DE_COMPRA A PEDIDOS,
'le decimos por tanto que el nodoPadre sea el elemento raíz. Vemos
'con un mensaje que tenemos dos nodos ORDEN_DE_COMPRA

Set nodoPadre = xmlDocument.documentElement

'enganchamos el nodo clonado a PEDIDOS, y se
colocará al final, 'ahora tendremos tres
ORDEN_DE_COMPRA. Lo comprobamos
'con un mensaje

nodoPadre.appendChild nodoClonado
MsgBox nodoPadre.xml

'borramos el nodo que habíamos clonado, volvemos
'a tener dos 'nodos ORDEN_DE_COMPRA. Lo
'comprobamos con un mensaje.

nodoPadre.removeChild nodoClonado
MsgBox nodoPadre.xml
End Sub
```

¡El mes que viene continuamos!

¡Saludos compañeros!



TECNICAS DE HIJACKING

por Vic_Thor

Este artículo posiblemente sorprenda a muchos y desespere a otros tantos, si, la vida es dura :)
Si has seguido todos los números de PC PASO A PASO podrás seguir este artículo, si no, ya sabes, a estudiar y a preguntar en el foro de Hack x Crack :)

Hola a tod@s, soy Vic_Thor, los que pasen por los foros de HackXcrack ya me conocen, aquellos que aun no lo hicisteis ya tenéis una tarea obligada, visitad:

http://www.hackxcrack.com/phpBB2/index.php por allí ando prácticamente todos los días y a parte de encontrarme a mi, hallaréis uno de los foros de mayor calidad y con mayor cantidad de información de lo que nunca os hubierais imaginado, totalmente en Español y con la gente más estupenda que he visto en todas mis andaduras en esto de la seguridad informática, acércate a los foros de HackXCrack, su valor se resume en una frase: Es un lugar donde preguntar dudas es obtener SIEMPRE una respuesta.

Actualmente se está desarrollando un Taller de TCP/IP, es una actividad única que no encontrarás en ningún otro sitio, desinteresada y con el único objetivo de escalar más allá del lugar donde otros se quedaron, no es necesario que tengas conocimientos previos, nuestra misión es empezar desde menos 1.

Este artículo está "robado" del Taller de TCP/IP, por su temática, contenido e interés, hemos pensado que el mejor lugar para explicar cómo llevar a cabo esta técnica, es un medio impreso como es esta Revista, que al igual que los foros de HackXcrack, es "otro animal único en su especie", sencilla, amena, profunda y que persigue una sola meta: EL CONOCIMIENTO.

Puedes seguir el Taller de TCP/IP en los foros de HackXcrack desde aquí:
http://www.hackxcrack.com/phpBB2/viewtopic.php?t=10306

En ese hilo de nuestros foros encontrarás también una pequeña explicación de cómo configurar el esnifer que utilizaremos en este artículo, si lo deseas puedes bajar ese mismo documento en:

http://www.iespana.es/vmthor/ForoCanal/Taller/Tcpip/Commview/Doc/CommView.zip

No me enrollo más, que lo que hoy nos ocupa es un asunto largo de explicar y corto de implementar...

IP-Hijacking

Venga... sentaos bien, poneos cómodos y tranquilidad que hay para rato....

Ya conocemos que al "colocar" un esnifer en una red podemos escuchar el tráfico que "ronda" por la misma, al colocar nuestro esnifer favorito en un segmento de red y desde el punto de vista de un usuario con "malas" intenciones podemos capturar muchas cosas, contraseñas, nombres de usuarios, páginas web, etc.

Esto es un "ataque pasivo", la máquina donde corre **el esnifer es un receptor del tráfico**, que descubre lo que circula por la red aunque no vaya dirigida a él.

El IP-Hijacking al contrario que un sniffing de paquetes, es un ataque activo, con el **IP-Hijacking lo que hacemos es robar una conexión ya establecida por otro usuario y máquina contra un servidor.**

En una conexión activa, este tipo de ataques se suelen utilizar contra maquinas que utilizan métodos de autenticación de password de un solo uso (syskey, NTLM, etc..) y en los que los ataques de toda la vida no surgen ningún efecto porque las contraseñas están cifradas. Ya sabes que aunque se pueda romper ese cifrado no es inmediato, imagino que conocerás lo que cuesta craquear un hash de Windows por ejemplo.

Los ataques activos se basan en el sniffing de un flujo de paquetes para encontrar una serie de datos que nos van a servir para "suplantar**" a una de las maquinas que forma parte de la sesión que estamos escuchando**, como este tipo de ataques se basan en sniffing, no nos servirán para nada si el flujo que estamos escuchando esta encriptado de cualquier forma, es aquí donde el Hijacking toma un especial protagonismo.

Para poder llevar a cabo con éxito esta técnica debemos ser capaces y aprender a:

- Utilizar un esnifer de forma adecuada
- Comprender la problemática de Ip-spoofing (falsear o suplantar una IP que no somos)
- Conocer como se negocia y se establece una sesión "normal" TCP
- Enviar paquetes manipulados que secuestren una conexión que otro estableció legalmente

Un escenario habitual podría ser éste:

1º) Colocamos nuestro esnifer favorito en la máquina que ejecutará el ataque, ni tan siquiera ha de ser así, existen esnifers a los que nos podemos conectar remotamente, como si se tratase de un troyano, pero eso puede quedar para otro artículo

2º) **Un servidor**, puede ser una BBDD un Controlador de Dominio, un Servidor Web, etc., en este ejemplo y para simplificar el asunto, **se trata de un Servidor Telnet con Autenticación NTLM** al que sólo puede acceder un determinado número de usuarios, con determinadas IP's y a determinadas horas, en ese servidor **la IP del atacante NO ESTA AUTORIZADA** a iniciar la sesión y **además el usuario de la máquina atacante NO SABE EL LOGIN NI PASSWORD** para entrar en el servidor.

3º) **Un cliente**, autorizado por el Servidor Telnet, tanto el usuario como la IP del cliente son **parte de los equipos de confianza del servidor.**

Aclaraciones:

En este escenario observarás que las tres máquinas pueden estar en la misma Red, eso es una ventaja, pero no tiene por qué ser así, lo que si es importante es que podamos comprometer el segmento de red del cliente o del servidor, puesto que si no, no podríamos hacer el sniffing del tráfico.

Al tener "esnifado" el tráfico teóricamente podríamos "robarle" la sesión al administrador remoto del Servidor Telnet UNA VEZ HAYA ESTABLECIDO la conexión, de tal manera que en adelante, el Servidor "pensará" que nosotros somos el cliente autorizado

¿Qué le pasaría al verdadero cliente?

R: Pues además de darle un cuchufrucu porque le birlaron la sesión por la cara, su máquina podría mostrar errores de conexión (cosa que tampoco es para alarmarse, puede ser un simple error de conexión, algo que es bastante frecuente y que si no es muy repetitiva, no produce muchas alarmas)

¿Podría el Cliente reconectarse?

R: SI, pero a nosotros eso nos dará igual, para el servidor será otra nueva sesión, autorizada y establecida, de tal forma que el verdadero cliente seguirá "haciendo sus cosas" y nosotros "las nuestras"

¿Y si el server nos desconecta a nosotros o perdemos la conexión por otros motivos?

R: Pues que tendremos que volver a robarla, la sesión perdida no se podrá reutilizar porque el server la dará por finalizada.

Deberías de tener MUY CLARO como se negocia y establece una sesión "normal" TCP, lo del saludo de tres vías, las señales (Flags), puerto origen, puerto destino, Ip origen, Ip destino y lo más importante:

- Los números de secuencia y asentimiento

Para "los no iniciados" haré una pequeña, pequeñísima revisión de todo ello, muy simplificada y con determinadas abreviaturas que usaré para que sea más cómoda su lectura:

Una conexión TCP esta definida únicamente por cuatro parámetros:

- La dirección IP del emisor (el que inicia la conexión)
- La dirección IP del receptor (el que recibe la conexión)
- El puerto TCP del emisor
- El puerto TCP del receptor.

El mecanismo que utiliza TCP/IP para saber si debe o no debe aceptar un paquete esta basado en comprobar una

serie de valores en cada paquete, si estos valores son los esperados por el receptor, el paquete es valido, en caso contrario se rechazan.

- Todos los paquetes llevan dos números que los identifican:

- El mas importante en el número de secuencia o (NºSEQ), este número de 32bits indica, el numero de bytes enviados, cuando se crea una conexión, el primer numero de secuencia que se envía se genera de forma aleatoria, es decir el numero de secuencia del primer paquete en una conexión no es 0, este número de secuencia va aumentando al menos en una unidad con cada paquete que se envía, aunque lo normal es que aumente el número de bytes enviados en los paquetes de datos y que aumente uno en los paquetes de control (flags)

- El otro número ligado al numero de secuencia, es el número de asentimiento o (NºACK), tanto el cliente como el servidor almacenan en este campo el valor del numero de secuencia siguiente que esperan recibir.

Por cada nueva sesión se generan **nuevos y diferentes números de secuencia**, para evitar que dos sesiones simultáneas tengan la misma serie de identificadores.

Aparte de los números SEQ/ACK (secuencia y asentimiento) la cabecera de un paquete TCP contiene otros campos, que por el momento no nos preocuparemos de ellos.

¿Cómo se establece una conexión? (Ejemplo de....)

Apertura de una conexión SIN INTERCAMBIO DE DATOS, es decir, sólo la negociación del saludo de tres vías.

Al principio, la **conexión por parte del cliente** esta cerrada (**CLOSED**) y **en el lado del servidor esta en estado de escucha (LISTEN)** en espera de nuevas conexiones.



Abreviaturas

Nº_SEQ_CLI	Nº secuencia generado por el cliente
Nº_SEQ_SRV	Nª secuencia generado por el servidor
Nº_ACK_CLI	Nº Asentimiento generado por el cliente
Nº_ACK_SRV	Nº Asentimiento generado por el servidor

1º) El cliente manda el primer paquete y le dice al servidor que sincronice números de secuencia con el **Flag SYN**:

Primer paquete que envía el cliente:

N_SEQ_CLI = aleatorio

Nº ACK_CLI = normalmente es cero

FLAG = **SYN**

Estado de la conexión: SYN-SENT

2º) Cuando el servidor recibe este primer paquete fija su primer número de ACK igual al número de secuencia que recibió del cliente que le acaba de llegar y establece el **flag SYN-ACK** y genera su propio número de secuencia que le devolverá al cliente

Primer paquete que enviará el servidor

Nº_SEQ_SRV = **aleatorio**

Nº_ACK_SRV = **N_SEQ_CLI + 1**

FLAG = **SYN+ACK** (comienza la sincronización de números de secuencia)

Estado de la conexión: SYN-RECEIVED

3º) Cuando el cliente recibe este paquete empieza a reconocer la serie de números de secuencia del servidor:

Paquete que envía el cliente y próximos números de SEQ y ACK que espera recibir el servidor

Nº_SEQ_CLI = **Nº_ACK_SRV + 1**

Nº_ACK_SRV= **Nº_SEQ_SRV + 1**

Estado de la conexión: ESTABLISHED

Cuando el servidor reciba este paquete sabrá que se ha establecido una nueva conexión y ambos, cliente y servidor, tendrán los datos suficientes para empezar a intercambiar paquetes de forma fiable.

Ejemplo:

1º) El cliente desea comunicarse con el Servidor, para ello:

A envía **un flag SYN**

Número de secuencia (aleatorio) = 2092

Asentimiento =0

2º) El servidor recibe ese paquete y envía:

Flag SYN+ACK

Número de secuencia (aleatorio) 0143

Asentimiento = 2093 (2092, que recibió del cliente + 1)

3º) El cliente lo recibe y le envía:

Un flag ACK

Número de secuencia = 2093 (que es el asentimiento del servidor)

Asentimiento = 0144 (0143, que es el nº de secuencia del servidor + 1)



¿Qué pasa cuando cliente y servidor se intercambian datos?

R: Pues que los números de secuencia y asentimiento se incrementan como antes, excepto que no será de uno en uno, sino que **se incrementarán en tantas unidades como bytes transmitidos**.

Además, cuando un host desea enviar datos a otro, el campo **flag se establece como PSH+ACK**

¿Cómo se cierra una conexión?

Una conexión se puede cerrar de dos formas:

a) Enviando un paquete con el **flag FIN activo**

b) Enviando un paquete con el **flag RST activo**

Si es el Flag FIN el que se activa, el receptor del paquete se queda en un estado de espera **CLOSE-WAIT y empieza a cerrar la conexión**

Si es el Flag RST el que esta activado, el receptor del paquete cierra la conexión directamente y pasa a un estado **CLOSED liberando todos los recursos asociados a esta conexión**

Pero no queda aquí todo....

Todo paquete IP lleva un número de identificación único que lo distingue de otros, además ese número de identificación puede ser usado tanto por el servidor como por el cliente para "seguir" la pista en caso de que se produzca un fenómeno especial en las comunicaciones, la fragmentación.

Si por cualquier motivo, el paquete se ha de fraccionar, para poder reensamblarlo en el destino será preciso reconstruirlo tomando en cuenta el ID de cada paquete y así poder agrupar "esos trocitos" en el destino.

El ID es fácil de calcular, puesto que **se incrementa en una unidad por cada paquete que envía un host a otro,** es decir que si se envió como ID 6896, el siguiente será 6897 y así sucesivamente por cada paquete independientemente del número de bytes trasmitidos....

El Ataque IP-Hijacking en nuestro ejemplo

El ataque IP-hijacking consiste en hacer creer al Servidor Telnet que esta manteniendo una conexión con un cliente autorizado y que los paquetes que esta enviando esta maquina no son validos

Por el contrario, los paquetes que vamos a enviar nosotros (el atacante) SÏ son validos, de esta manera nos apoderamos de una conexión

¿Qué le ocurre al Servidor?

R: Le parece todo normal

¿Qué le ocurrirá al Cliente Autorizado?

R: Pensará que el servidor le ha cerrado la conexión por cualquier razón

¿Qué tenemos que modificar y/o enviar al Servidor?

R: Para poder secuestrar la conexión establecida entre el cliente autorizado y el servidor telnet, tenemos que ser capaces de hacer creer al servidor Telnet que los paquetes del cliente autorizado ya no son válidos.

¿Cómo puede pensar el servidor Telnet que los paquetes del cliente autorizado ya no son válidos?

R: Lo que vamos a hacer es que los números SEQ/ACK (secuencia/asentimiento) que envía el cliente sean erróneos y entonces el servidor los descartará.

Si... pero.... ¿Cómo?

R: Enviaremos datos adicionales a los paquetes que envía el cliente destinados al Servidor spoofeando la IP del Cliente

¿Y con esto qué se consigue? ¿Explica mejor eso de paquetes adicionales?

R: Cuando el Servidor Telnet reciba esos paquetes que aparentemente los envió el Cliente, actualizara sus números ACK, y aceptara estos datos modificados, con los nuevos números SEQ/ACk que nosotros hemos forzado

Sigo sin entenderlo, veo las intenciones pero no comprendo en qué afectará eso al cliente

R: A partir de este momento, todos los paquetes que envíe el cliente serán rechazados, ya que esta utilizando los SEQ/ACK antiguos, sólo nosotros estaremos en poder de los nuevos....

Aja, estupendo... ¿y luego qué?

R: Una vez que hemos logrado esto, ya esta hecho, ya tenemos el control de la conexión, lo único que tenemos que hacer ahora es calcular los números SEQ/ACK de cada paquete que enviemos para que corresponda con lo que espera el servidor.

La última respuesta es "engañosa" porque eso de que "lo único que tenemos que hacer es calcular ...." puede ser una auténtica pesadilla, hay que conocer "al dedillo" los protocolos utilizados, en este caso TCP, IP y Telnet, pero imagina lo que sería suplantar una sesión HTTP o SMB, ufff un horror, pero se puede, no lo dudes.

Siendo "malvados" también podríamos cerrar todas las conexiones que nos de la gana, en lugar de enviar paquetes que "continúen" la conexión, enviamos paquetes FIN o RST con los números de secuencia válidos y chas... que luego el Cliente lo vuelve a intentar... pues otra vez... y chas de nuevo.... y así hasta que uno de los dos se canse, seguramente el Cliente se pondrá de los nervios.

Ahora nuestro ejemplo detallado

La práctica que vamos a realizar es la siguiente, vamos a secuestrar una sesión Telnet establecida por un cliente autorizado y le pediremos al servidor que nos transfiera un archivo a nosotros, si se realiza con éxito pasará todo esto:

El servidor hará lo que le pedimos y nos transferirá el archivo (suponemos que el archivo que nos interesa recibir es uno llamado sc.exe que está en el directorio "por defecto"

El cliente autorizado perderá la conexión que él estableció legalmente

Nosotros recibiremos el archivo del servidor

Para ello debemos preparar el escenario:

En nuestro **equipo atacante (172.28.0.25)** montamos un **esnifer y un servidor TFTP**

El equipo "de confianza" (172.28.0.50) inicia una sesión telnet contra un servidor

El Servidor Telnet (172.28.0.9) que SOLO acepta la conexiones del equipo de confianza

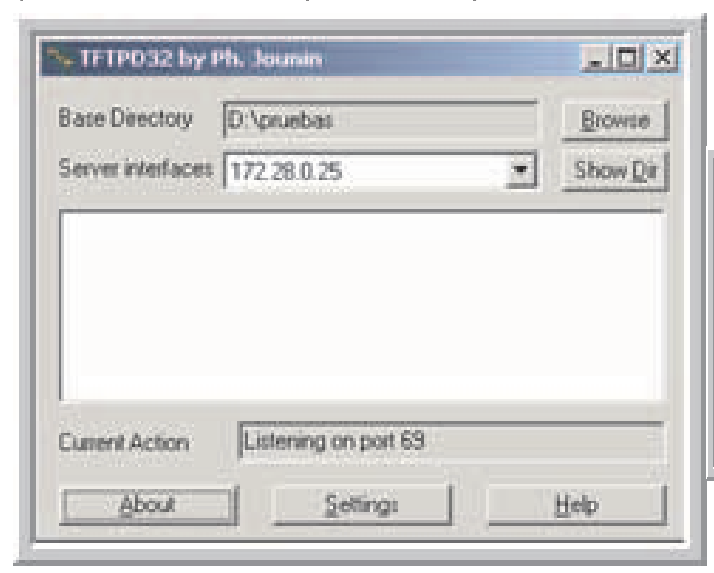
Resulta que nos encontramos escuchando esa sesión telnet establecida entre los equipos cliente y servidor (172.28.0.50 y 172.28.0.25) desde nuestro equipo atacante (172.280.0.25), pero no podemos iniciar sesión telnet contra el server puesto que ni tenemos password ni login, y **aun en el caso de que lo averigüemos el Servidor Telnet rechazaría la sesión porque no somos una IP válida para él.**

Entonces, **vamos a secuestrar la conexión** que inició el equipo de confianza, nos haremos pasar por él **y le pediremos al server que nos transmita un archivo a nuestro equipo!!!!!**

Lo primero es activar servidor **TFTPD32** en nuestro equipo, ya sabes utilizaremos el tftd32.exe que nos enseñó la Revista en sus primeros números, si no dispones de él puedes bajarlo de:

http://www.hackxcrack.com/DELTA/tftpd32e.zip

Lo único que tienes que hacer es modificar "Base Directory" para que apunte al directorio donde quieras recibir el fichero que transferirá el server (Botón Browse).



Luego ponemos en marcha nuestro esnifer y filtraremos el tráfico telnet (puerto 23 de TCP) para escuchar "la conversación" entre el servidor y el cliente autorizado.

Pongamos que nos encontramos algo parecido a esto....

No	Protocolo	Direcciones Físicas	Direcciones IP	Puertos	Difer...
1	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	11,116
2	IP/TCP	Compaq <=> Toshiba	172.28.0.9 <=> 172.28.0.50	23 <=> 1088	0,100
3	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	0,141
4	IP/TCP	Compaq <=> Toshiba	172.28.0.9 <=> 172.28.0.50	23 <=> 1088	0,100
5	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	0,010
6	IP/TCP	Compaq <=> Toshiba	172.28.0.9 <=> 172.28.0.50	23 <=> 1088	0,090
7	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	0,180
8	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	1,192
9	IP/TCP	Compaq <=> Toshiba	172.28.0.9 <=> 172.28.0.50	23 <=> 1088	0,100
10	IP/TCP	Toshiba <=> Compaq	172.28.0.50 <=> 172.28.0.9	1088 <=> 23	0,190

Hombre esto es poco significativo... se tratan de una serie de paquetes "pasantes" es decir, que nuestro esnifer capta y no van dirigidos a el propio equipo donde está corriendo (fíjate que el símbolo es < = >, entran y salen de las ip's 172.28.0.50 y 172.280.9 pero ninguna va dirigida exactamente a nosotros

Tampoco sabemos exactamente de qué se trata, solo podemos observar que los puertos a los que el cliente (Toshiba) se conecta con el servidor (Compaq) es el 23.

También vemos el puerto que usa el cliente, el 1088, puerto dinámico y que parece ser que están haciendo "algo"

Vamos a "escudriñar los tres últimos, podríamos hacerlo de todos, pero para no aburrir.....

De todo ello **nos vamos a fijar en unas cuantas cosas: Identificación, Longitud total del Paquete IP, Nº de secuencia, número de Ack , Bandera-señal flag y Bytes enviados**

Una tablita mejor:

Host Emisor	ID. IP	Longitud total	Nº SEQ	Nº ACK	Flag	Nº de Bytes de datos
Toshiba a Compaq	40674	41	217092584	622309261	PSH+ACK	1 byte
Compaq a Toshiba	34085	57	622309261	217092585	PSH+ACK	17 bytes
Thosiba a Compaq	40675	40	217092585	622309278	ACK	0 bytes

Columna Identificación:

El Toshiba incrementa en una unidad el campo Id. Por cada paquete que envía

El Compaq también lo hace, sólo que como no tenemos más que una entrada no podemos compararlo, pero en caso de que Compaq iniciase una nueva transmisión el valor **ID.IP** de la misma sería 34086 para el ejemplo de la tabla anterior....

Longitud Total,

Esto vale para cualquier dirección que tome el paquete y para todos los paquetes

Suma de 20 bytes de encabezado IP+ 20 bytes de encabezado TCP+ nº bytes de datos**Por qué sumas 20 de encabezados.... ¿siempre son 20?**

R: Pues en condiciones normales, SI. El paquete IP serán 20 bytes y TCP también a menos que se utilicen opciones, que no es el caso... por tanto para calcular la longitud total del paquete a enviar habrá que sumar 40 al número de bytes enviados

Nº Seq. Y ACK

El número de secuencia del último paquete se incrementa en tantos bytes como bytes de datos enviados desde el propio toshiba

217092585 = 217092584 + 1

También podemos calcular el Nº SEQ como el Nº ACK del paquete anterior recibido de Compaq... es lo mismo

El número ACK del último paquete se incrementa en tantos bytes como bytes recibidos

622309278 = 62230961 + 17

Cuando es el Compaq el que envía pasa lo siguiente:

El número de secuencia es el mismo número de ACK que envió Compaq (622309261)

El número de ACK es el número de secuencia que envío toshiba en el paquete anterior + 1

217092585 = 217092584 + 1

Buahhhh!!!! Menuda rallada, qué comedura de coco....

Vale, **pensemos en el próximo paquete que se debería enviar**....

Si Toshiba le envía un nuevo paquete a Compaq, por ejemplo un ACK con 0 bytes de datos

Id IP = Id Ip +1, es decir 40676 puesto que el último fue el 40675

Longitud total = 20 + 20 +0, es decir 40

Nº SEQ = 217092585 + 0, es decir 217092585

Nº ACK = 622309278 + 0, es decir 622309278

El flag sería un ACK

El número de bytes enviado cero

Y qué le debería responder Compaq.... pues lo mismo, respondería otro ACK como flag, intercambiando los números de secuencia por los valores del ACK y viceversa, de tal forma que el la próxima transmisión que haga toshiba utilizará el nª ACK como número de SEQ y el número de SEQ como número de ACK

Bien, **entonces enviemos el paquete correcto que ejecute el tftp....**

Toshiba envía a Compaq:

Id IP 40676 (se supone que no se envió ningún nuevo paquete ACK)

Longitud total = 20 +20 + 31 = 71

Nº SEQ = 217092585 + 0

Nº ACK = 622309278 + 0

Flag PSH+ACK

Nº Bytes: 31

Los 31 bytes son: tftp –i 172.28.0.50 put sc.exe



Tftp es un ...

Tftp es un cliente que los Windows 2000 y XP llevan "incorporados de serie", la sintaxis anterior significa lo siguiente:

Tftp (cliente para el servidor tftpd32 que tenemos a la escucha)

-i el archivo que deseamos transmitir es un archivo binario

172.28.0.50 es la dirección IP del equipo que recibirá el archivo (la del atacante que a su vez tiene a la escucha el servidor de TFTP)

put, le indica al comando tftp que lo que se desea hacer es una transeferencia de archivos desde el equipo hacia la IP indicada, si deseáramos "subir" un archivo desde nuestra máquina al servidor telnet, tendríamos que usar get en lugar de put

sc.exe, es la ruta y nombre de archivo a descargar., suponemos que la ruta es el "directorio actual" sino fuese el caso, habría que indicarla, por ejemplo: c:\BBDD\clientes\sc.exe

Si cuentas los caracteres incluidos los espacios, los puntos, las letras y números **suman 30....**

¿Por qué son 31 los bytes a transmitir entonces?

R: Porque faltaría el "enter" es decir, el carácter 0D en hexadecimal que equivale al retorno de carro....

¿No decías que el ACK que envía Toshiba se deben sumar los bytes recibidos? Entonces, **¿No sería 622309278 + 31?**

R: NO. Recuerda que son RECIBIDOS, y en el último paquete recibido de Compaq, se recibieron CERO bytes!!!

¿Y por qué no se incrementa en uno o en 31 el número de secuencia?

R: Porque sólo es así cuando RECIBIMOS un PSH+ACK y en este caso lo estamos ENVIANDO

Luego será Compaq quien recalcule los números de secuencia y ACK correctos y nos los devolverá en el siguiente ACK

En resumen, que cuando utilicemos esta técnica**, lo único que tenemos que hacer es:**

modificar la Identificación del ID de IP a ID de IP + 1

El tamaño total del paquete (40 + los bytes que enviemos),

cambiar el flag a PSH+ACK (valor 18 hexadecimal)

Recalcular el checksum IP

Recalcular el checksum TCP

Añadir los bytes de datos al final del paquete

Si todo fue bien, en el momento que enviemos ese paquete de datos:

Toshiba no sabrá que le han secuestrado la conexión, simplemente "se colgará" la sesión telnet que el estableció "legalmente"· Compaq recibirá "la orden tftp" y procederá a ejecutarla, con lo que nos transferirá el archivo deseado

Compaq enviará paquetes ACK a toshiba después de procesar el paquete que le enviamos desde el equipo atacante y toshiba responderá otros ACK pero no serán válidos puesto que estarán "anticuados", comenzará un "baile" de ACK's que envía uno y ACK's que envía el otro, como no se ponen de acuerdo en el número de secuencia y en el número de asentimientos ACK, la sesión terminará por quedarse colgada y le aparecerá un mensajito a Toshiba como que se ha perdido la conexión con el host.... **pero nosotros habremos RECIBIDO EL FICHERO EN NUESTRO SERVIDOR TFTP**

Mira esta pantalla de lo que ocurrió al enviar el paquete...

Contrasta la información del paquete enviado con la de nuestro ejemplo....

ID	40676
Total Length	71
Sequence:	217092585

Acknowledgement 622309278

Flags PSH+ACK

Datos (sombreado) tftp –i 172.28.0.25 put sc.exe (y un carácter 0D de retorno de línea <CR>)

A ver... lo del círculo rojo... **¿Por Qué aparecen 85 bytes como tamaño del paquete en las líneas inferiores del generador? ¿No habíamos quedado que eran 71?**

R: Porque los otros 14 bytes corresponden al direccionamiento MAC, 6 para la MAC destino, 6 para la MAC origen y 2 bytes para el protocolo siguiente, en este caso IP.

¿Y qué pintan las MAC's aquí?

R: Pues sencillo, para que pueda existir direccionamiento lógico (direccionamiento por IP) es obligatorio que exista direccionamiento físico (direccionamiento por MAC) si no hay MAC origen, MAC destino y tipo de protocolo a usar NO HAY COMUNICACIÓN.

¿Siempre son 14 bytes el direccionamiento MAC?

R: Para redes basadas en Ethernet SI. Siempre son 14 y en ese orden, además como la transmisión es TCP/IP el campo tipo siempre será 08 00, lo que cambiará en cada caso serán las direcciones MAC

Si analizamos los primeros 14 bytes de la captura que tomó el esnifer, verás que son estos valores:

00 10 60 5C 6B 13 00 00 39 C1 6A C2 08 00

Dónde:

00 10 60 5C 6B 13 corresponde a la dirección **MAC del destino** (El servidor Telnet)

00 00 39 C1 6A C2 corresponde a la dirección **MAC del origen** (el cliente autorizado)

08 00 Corresponde al protocolo a utilizar.... **08 00 es IP**

Entonces... además de falsear la IP, predecir los números de secuencia y asentimiento, averiguar los puertos de conexión, etc... ¿Debo falsear también la dirección MAC?

R: SI. Efectivamente, para que el ataque tenga éxito, además de todo lo anterior será preciso suplantar la dirección MAC del equipo

Recordatorio:

Todos los ordenadores guardan en memoria una tabla que relaciona las direcciones IP con las direcciones MAC de los otros ordenadores a los que se conectan, esa tabla puedes verla si ejecutas el comando arp –a desde la shell del sistema.

En el caso de que un equipo no conozca la dirección MAC de otro con el que quiere comunicarse, lo que hará es enviar una petición broadcast (una consulta a todos los equipos de la LAN) algo así como si un host pregunta a todos ¡¡¡ el que tenga la dirección Ip xxx...xxx.xxx.xxx.xxx que me diga cual es su MAC, por favor!!!

Si la IP xxx.xxx.xxx.xxxx existe en la red, le entregará su dirección MAC al ordenador que la pidió y éste último actualizará su tabla de direcciones MAC en su memoria para posteriores usos y así no tener que andar preguntando lo mismo cada vez..., si no existiese ese equipo en la LAN no se podrían comunicar y el paquete de datos no saldría del host emisor....

Por tanto tenemos que los primeros 14 bytes en una transmisión Ethernet son:



Donde el **campo tipo puede ser** uno de los siguientes valores

Tipo	Siguiente protocolo a utilizar
08 00	IP versión 4
08 06	IP ARP
80 35	IP RARP
08 08	Frame Relay ARP
86 DD	IP version 6
08 05	X 25

¿Cómo se construye un paquete TCP/IP al enviarse por una red Ethernet?

R:

- **Primero** se añaden 14 bytes que corresponden a la **cabecera MAC** como acabamos de ver
- **Segundo** se añaden 20 bytes para la **IP**
- **Tercero** de suman otros 20 bytes para **TCP**
- **Y por último** se pegan los **datos a enviar**

Bueno, no siempre es así exactamente, pero para el caso que nos ocupa puede servir.

Hay que reseñar que **esos últimos bytes** (los datos a enviar) **pueden pertenecer a su vez a otros protocolos**, en nuestro caso pertenece al protocolo TELNET, pero podría ser una petición Web (HTTP), una sesión NetBIOS, un FTP, etc...

A esto se le llama **ENCAPSULACIÓN**, cuando se forma el paquete se van añadiendo las cabeceras de los protocolos que intervienen en la comunicación junto con sus datos, al final, tendremos un único paquete con tantos bytes como bytes de datos a enviar + las cabeceras de los protocolos que los transportan.

Como entenderás hay muchas variantes, puede ser UDP en lugar de MAC+IP+TCP, puede ser ARP, etc... todo llegará, por el momento prestaremos atención a las cabeceras y campos que necesitamos modificar para que nuestro IP-Hijacking tenga éxito.

¿Qué campos tenemos que modificar de la cabecera IP?

R:

- Total Length (Longitud total del paquete) que será igual a 20 de IP + 20 de TCP + nº bytes enviados
- ID. IP, Identificador de paquete, que será el valor que exista + 1
- Source IP, la dirección IP del emisor, como se trata de suplantar la IP de un equipo diferente, obviamente no debe ser la nuestra sino la del equipo a falsificar... IP-Spoofing
- Destination IP, la dirección Ip del receptor, vamos la del servidor Telnet de nuestro caso
- Checksum, mmm esto parece ser complicado... veamos el checksum es un valor que asegura que la cabecera IP no ha cambiado por errores de transmisión u otros motivos, precisamente este es uno de los escollos a salvar. Sin embargo el campo de verificación checksum no es un mecanismo de seguridad ante este tipo de ataques, simplemente es un medio que utiliza IP para asegurarse de que lo que se envió es precisamente lo que se recibe en el destino y que no sufrió alteraciones. Es una suma de comprobación.... no te preocupes, cuando llegue el momento, nuestro esnifer será capaz de generar un checksum correcto para que no haya problemas

¿Tendremos que alterar algunos valores de la cabecera TCP?

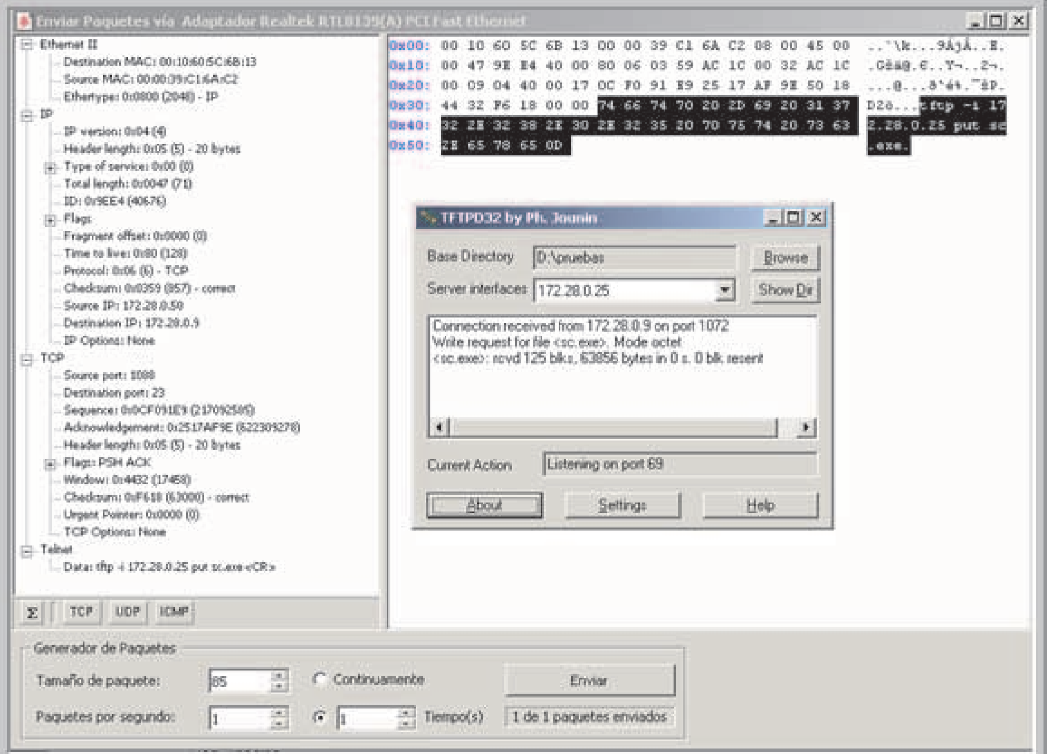
R: **SI**, los que nos interesan son:

- Source port, Puerto origen por el que el host emisor se comunica con el receptor, en nuestro caso, como se trata de una sesión iniciada por un cliente Telnet, será un puerto dinámico (entre 1024 y 5000 para Windows)
- Destination port, puerto destino por el que el otro extremo escucha, en nuestro ejemplo corresponde al puerto 23 de TCP (TELNET)
- Sequence: Es el número de secuencia, que se calculará tal y como vimos anteriormente.
- Acknowledge: Será el número de asentimiento, idem del anterior.
- Checksum, lo mismo que en IP pero aplicado a TCP, habrá que recalcularlo....
- FLAGS, será un único byte que le dice a TCP si lo que se envía es un SYN, ACK, PSH, FIN, RST, etc... los valores para los flags que necesitamos son:

- FIN → 01
- SYN → 02
- RST → 04
- ACK → 10
- SYN+ACK → 12
- PSH+ACK → 18

Puede haber más pero con estos nos sobran para lo que buscamos….

Te podré la misma pantalla que antes pero toda completa para que veas que existe el encabezado MAC.



En el centro pegué la captura de pantalla del servidor TFTPD32, observa como se recibieron sin problemas los datos de la ip 172.28.0.9 a la cual NUNCA, NUNCA, NUNCA nos conectamos.

A todos los efectos quien hizo "algo irregular" fue el equipo 172.28.0.50

El equipo atacante sólo aparece como "receptor" en el segmento de datos, en las conexiones establecidas con el servidor SOLO APARECERÁ la IP del equipo de confianza.

Poco se puede hacer contra esto, aunque se "cifre" las sesiones Telnet o cualquier otro tráfico, siempre puede aparecer un "desalmado" que secuestre la sesión prediciendo los números de secuencia y asentimiento que un servidor espera recibir de sus clientes, en el ejemplo es sencillo puesto que el atacante está en el mismo segmento de red que cualquiera de las otras dos máquinas, si no fuese así....

¿Se podrá hacer?

R: **SI**. Pero sería un ataque "a ciegas", además de predecir los números de secuencia y ACK correspondiente, tendremos que "imaginar" que es lo que ocurre, eso es el llamado **Blind Spoof**, técnica complicada donde las haya, pero no imposible.

Dentro de poco a programar sockets en C, si somos aplicados podremos crearnos nuestros propios programas para hacer esto mismo, no creas que es difícil, ni mucho menos, lo más difícil es saber cómo hacerlo y eso lo acabas de aprender. Esperaremos con impaciencia los cursos de el_chaman para empezar a enviar paquetes mediante generadores propios...

Una buena práctica sería un programa que "rastree" el último ACK recibido y envíe un RST a la dirección origen... una lamerada ya lo sé... pero bien montado dejas a cualquier máquina sin posibilidades de conectarse a nada, cada vez que inicia una conexión.... nuestro "virus" envía un RST y la cierra.... para volverse loco....

Llegó el momento...

Veamos cómo se puede generar ese paquete "tan ansiado", damos por supuesto que ya tenemos preparado y corriendo en nuestra máquina el programa TFTPD32 que se ocupará de recibir el archivo que nos interesa robar del servidor....

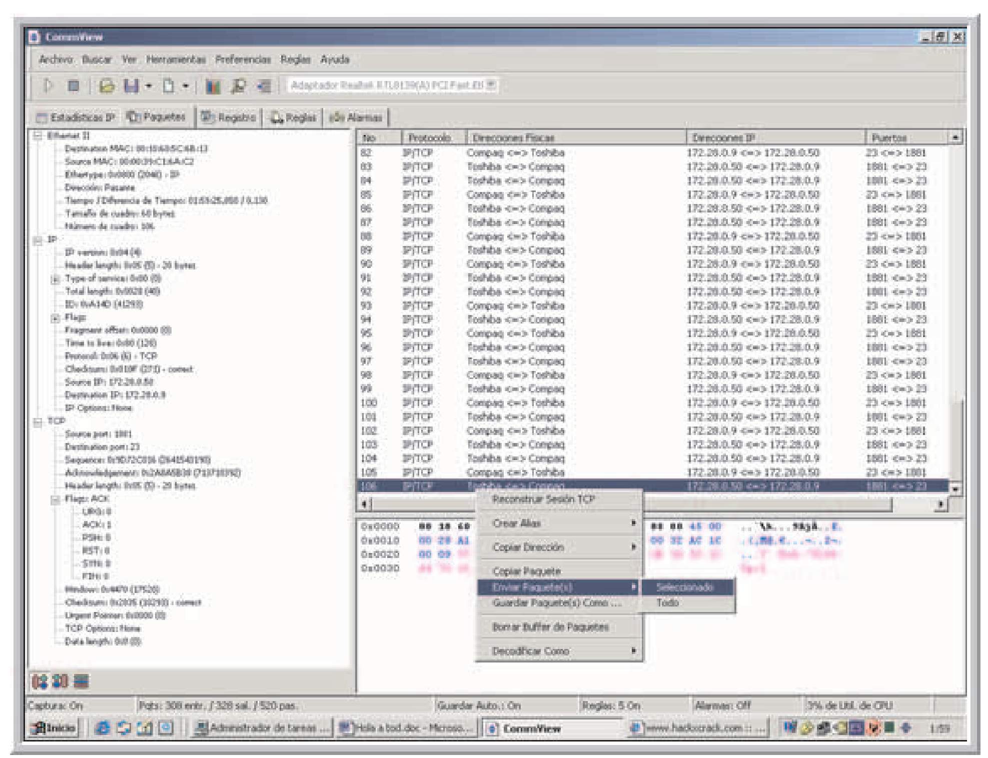
1º) **Iniciamos** el esnifer CommView

2º) Configuramos los filtros necesarios para escuchar nada mas que "las conversaciones" en las que interviene el **puerto 23 de TELNET**, con esto nos evitamos capturar otro tipo de tráfico innecesario para el ejemplo

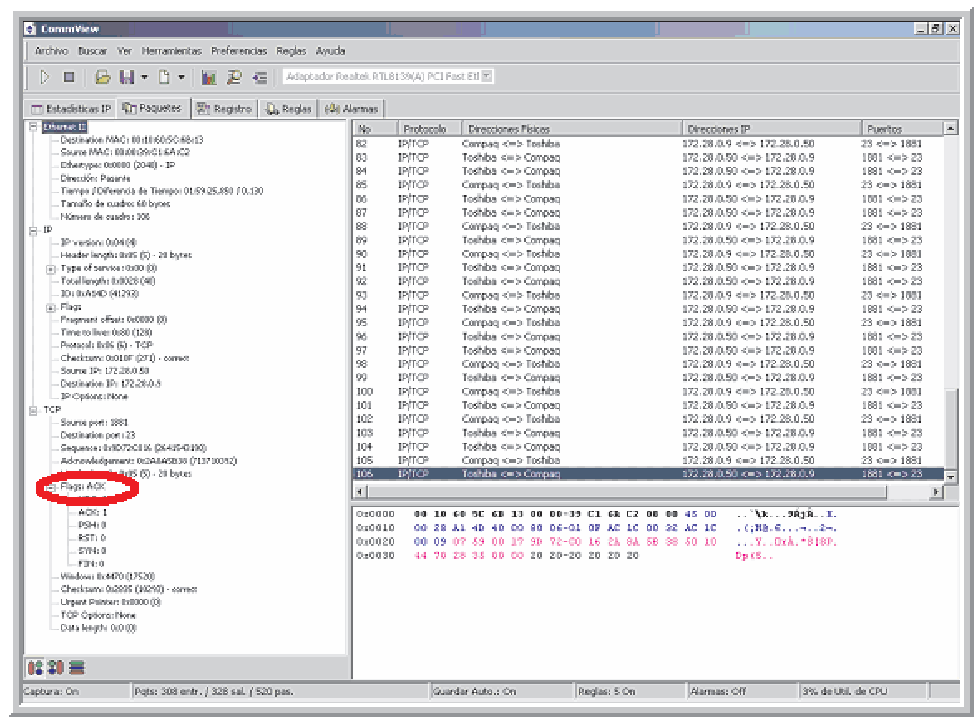
3º) **Esperamos pacientemente** a que la víctima y servidor

establezcan una sesión Telnet, o a lo mejor, ya está establecida y están charlando "amigablemente", pongamos que esto último...

4º) Sabemos que la IP del Servidor telnet es la 172.28.0.9 y la del cliente la 172.28.0.50, así que lo que tenemos que hacer es **localizar el ULTIMO paquete de datos con flag ACK** que servidor y cliente se intercambiaron, no será muy difícil porque será el último que capturemos, lo que debemos es asegurarnos que sea un ACK y no un FIN o un RST, puesto que en estos casos significaría que cliente y servidor pusieron fin a su "charla" y nos quedaremos "compuestos y sin novio"



5º) **Atención.... nuestro esnifer capta algo....** veamos....



Fijamos el resalte en el último paquete y nos aseguramos que se corresponda con un Flag ACK

Sobre ese paquete pulsamos el botón derecho del ratón y seleccionamos Enviar paquete-seleccionado

Aparecerá esta pantalla(***ver imagen 1*****): En tamaño del paquete seleccionamos 85 bytes** (ya lo vimos antes, que son:

- 14 para la cabecera MAC
- 20 para IP
- 20 Para TCP
- 31 de Datos a inyectar

Lo que tenemos que modificar son las zonas redondeadas en rojo (IP) azul (TCP) y verde (Datos)

¿Por qué sabes que son esos bytes y no otros?

R: CommView es "inteligente" cuando marcamos un campo de la zona izquierda, por ejemplo total Length, el nos resaltará los valores hexadecimales a que corresponde dicho campo, así que lo único que hay que hacer es ir pinchando en los campos a modificar e ir sustituyendo los valores que existen por los nuevos...

¿No se modifican las IP, ni las MAC?

R: NO. Las que están son correctas, observa que se trata

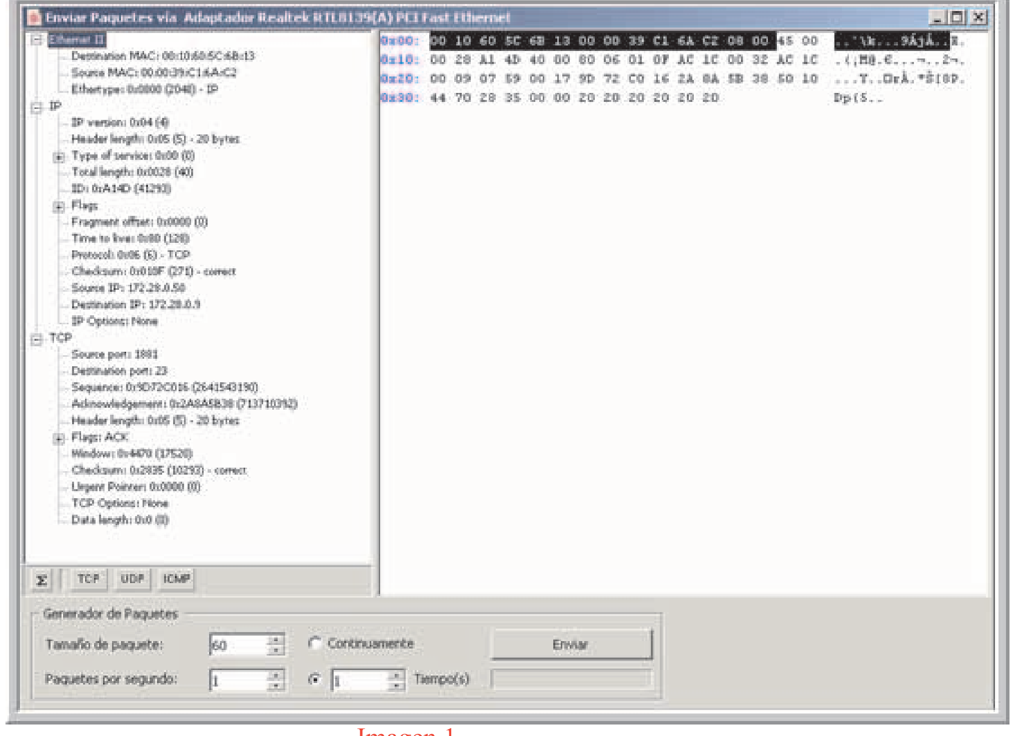


Imagen 1

de un paquete que supuestamente enviará el equipo de confianza al servidor Telnet, como es un paquete que nuestro esnifer capturó, los valores correctos ya están puestos, no es necesario modificarlos.

¿Y los números de secuencia y asentimiento?

R: Tampoco es preciso modificarlos, el último paquete que envió el Cliente de Confianza al Servidor Telnet es un ACK, es decir, un asentimiento de los datos recibidos, por tanto los números de secuencia y asentimiento que espera el Servidor son precisamente los que acaba de enviarle. Recuerda que un Flag ACK no incorpora datos, al no haber bytes a transmitidos, los números de secuencia y asentimiento no varían.

¿Y los puertos?

R: Tampoco, por el mismo motivo que antes, se trata de un paquete que envía el Cliente al Servidor, tanto el puerto destino como origen son correctos.

__Resumiendo__, lo único que tengo que **modificar** son: Total Length, ID.IP, Flags TCP y los datos al final del paquete *(ver imagen 2)*
__Pues vamos a ello:__

Total Length había 0028 (40 decimal) **tendremos que poner 0047** (85 decimal)

Id.IP tenía el valor A14D (41293 decimal) **pondremos A14E** (41294 en decimal)

Flags TCP tenía 10 (ACK) pondremos 18 (**PSH+ACK**), porque vamos a enviar datos.

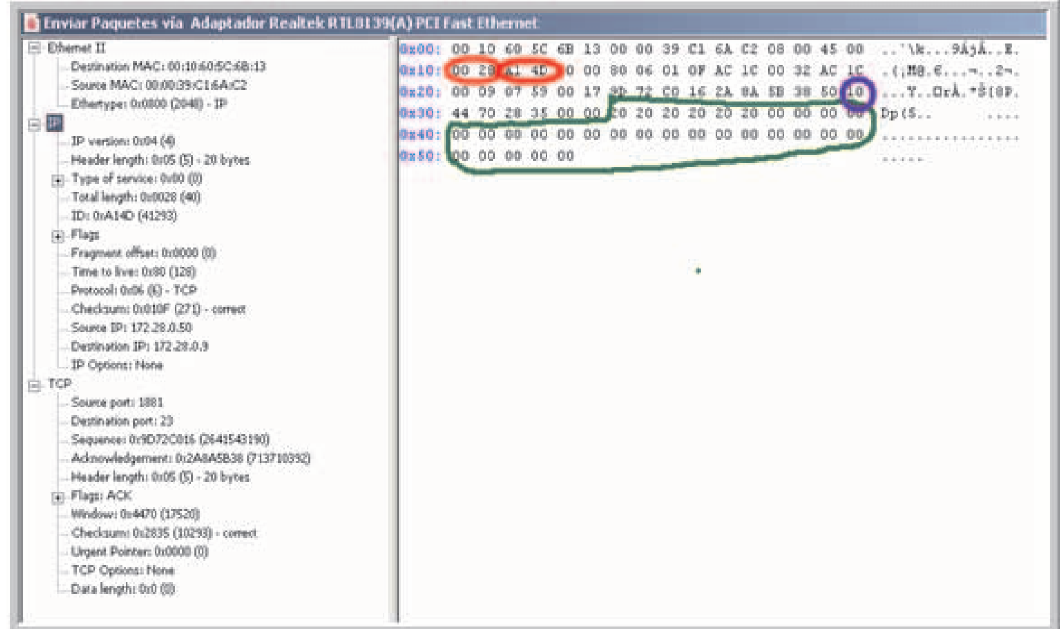


Imagen 2

Observa que al **haber aumentado el tamaño a 85 bytes,** apareció al final del paquete de datos una **nueva entrada que pone Telnet**, CommView interpreta que al enviarse el paquete hacia el puerto 23 será unasesión telnet, listo verdad?

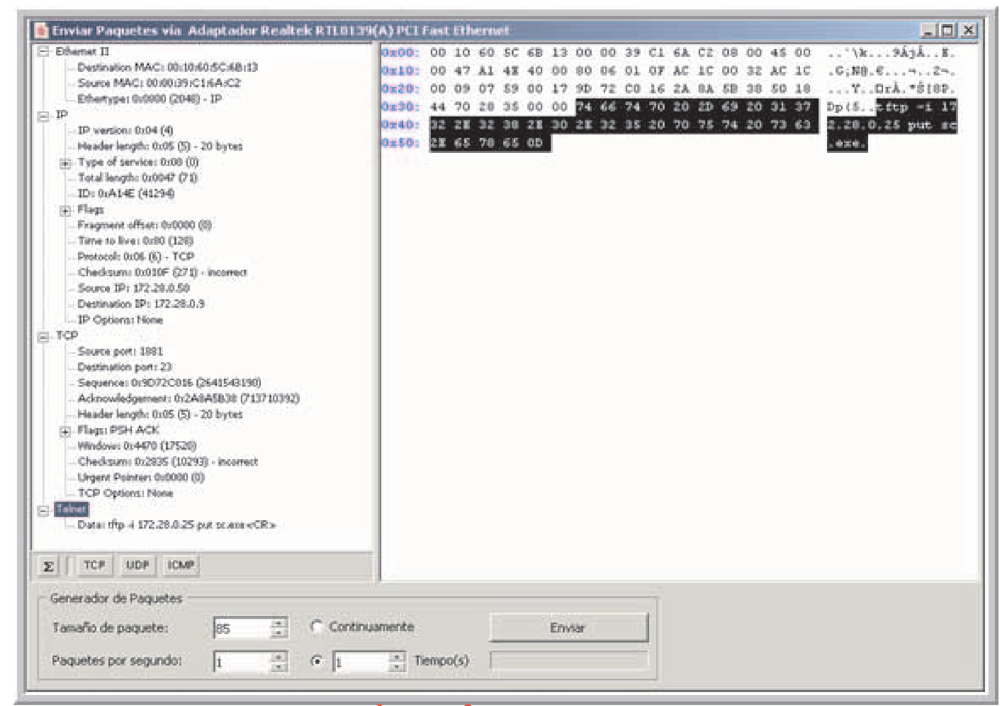


Imagen 3

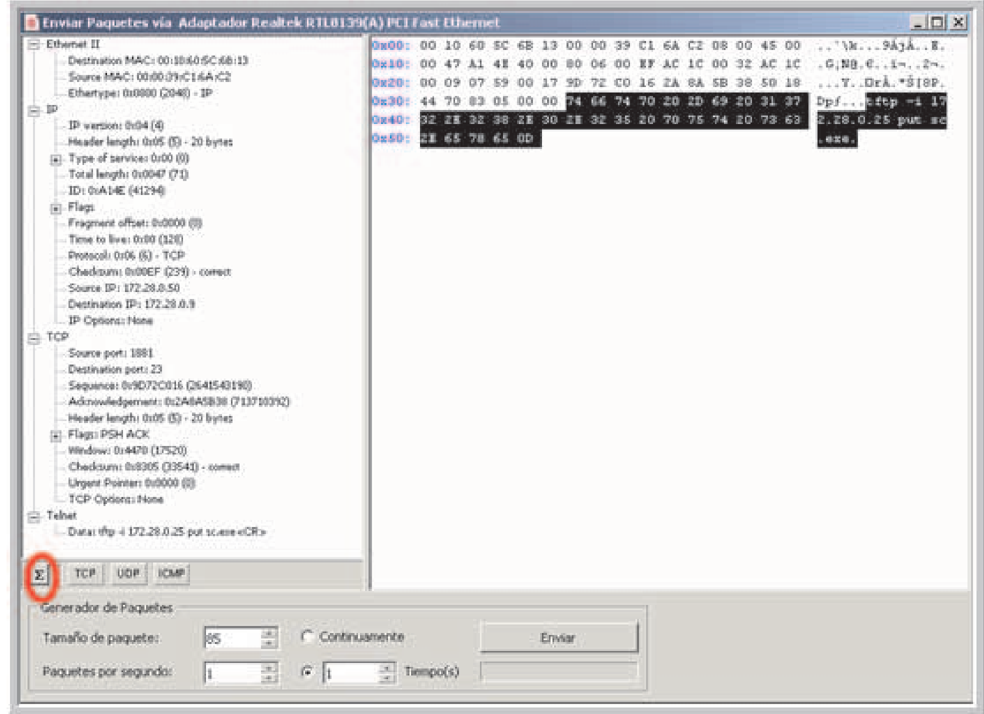


Imagen 4

Los datos los podemos escribir en hexadecimal o en ASCII, lo más sencillo será pulsar en la zona de más a la izquierda de la parte hexadecimal y escribir directamente lo que queremos enviar, es decir, tftp –i...... eso será mucho más sencillo que convertir cada letra en sus respectivos valores hexadecimales

Al final debemos escribir un 0D en la zona hexadecimal, que se **corresponde con un "enter"** un <CR> **(ver imagen 3)**

Sin embargo no podemos enviar el paquete todavía... falta algo....

¿Recuerdas qué será?

R: Seguro que sí lo has hecho, los valores de los campos **CHECKSUM** tanto de TCP como de IP, si observas lo que te dice CommView, pone que son "incorrect"

Es lógico que lo sean, acabamos de modificar bastantes valores del paquete y por tanto los checksum deben volver a recalcularse, ufff complemento a uno.... bueno de eso se ocupará el propio Commview, veamos como.

En un círculo rojo te remarqué un botón "especial", el símbolo del sumatorio en muchas hojas de cálculo, si pinchas en él, Commview calculará los checksum correctos para el paquete y estará listo para enviar...**(ver imagen 4)**

Fíjate que **tras pulsar en el signo de sumatorio**, los campos checksum tomaron el valor "correct", **AHORA SI PUEDES DARLE A ENVIAR**

Una vez inyectado este paquete, el equipo de confianza perderá la conexión con el Servidor Telnet y éste último ejecutará la ordenque le enviamos, un tftp, en nuestro equipo recibiremos sin problemas el archivo **y FIN DE LA PARTIDA.**

Como verás "hacerlo a mano" es laborioso, no imposible, pero demasiado entretenido cuando una sesión está muy activa... mientras enviamos el paquete, lo modificamos, recalculamos checksum, etc, es probable que nuestro paquete quede "anticuado" debido a nuevas peticiones del Cliente de confianza., vamos que hay que darse mucha prisa si lo deseamos hacer manualmente, pero a grandes males, grandes remedios...

Para LiNUX existe una utilidad llamada Hijacking.c, es un programita en C que hace todo esto solito, secuestra la sesión y ejecuta los mandatos que le metamos desde la línea de consola, **si además lo combinamos con un esnifer que nos informe de los números de secuencia y asentimiento junto con los estados de las señales Flag TCP**, secuestrar la sesión será un juego de niños... te recomiendo para ese objetivo el cóctel **Hijacking.c + snort = Robo y secuestro garantizado**

Para Windows no conozco herramientas que automaticen todo esto... ya sabes, Vic_Thor, un motivo más para ir cambiándote a LiNUX

Espero que os haya gustado la práctica, algunos de vosotros estaréis pensado que si esto sólo se puede hacer en LAN... pues no... **también es posible trasladar el escenario a Internet**, pero en esa red la guerra es constante, existen o pueden existir muchas barreras que lo dificulten, desde nuestro propio router, hasta los routers de nuestro proveedor, pasando por NAT, IDS, firewalls, etc... pero se puede, sí señor, claro que se puede, sino... ¿Cómo piensas que el ínclito Kevin Mitnick asaltó el Xterminal del Sr. Shimomura?

Pues más o menos como te lo he contado, también hay que advertir que en esa época no existían tantos mecanismos de seguridad como los hay hoy en día, pero el spoofing e Hijacking siguen siendo vigentes, por no decir cuando es un equipo de la LAN el que se conecta a un equipo de Internet remotamente... en esos casos Hijacking es una técnica devastadora.



NÚMERO 3:

-PROXY: OCULTANDO NUESTRA IP.
ASUMIENDO CONCEPTOS.
-PROXY: OCULTANDO NUESTRA IP.
ENCADENANDO PROXIES.
-PROXY: OCULTANDO NUESTRA IP.
OCULTANDO TODOS NUESTROS PROGRAMAS TRAS LAS CADENAS DE PROXIES.
-EL SERVIDOR DE HACKXCRACK: CONFIGURACIÓN Y MODO DE EMPLEO.
-SALA DE PRACTICAS: EXPLICACIÓN.
-PRÁCTICA 1ª: SUBIENDO UN ARCHIVO A NUESTRO SERVIDOR.
-PRÁCTICA 2ª: MONTANDO UN DUMP CON EL SERV-U.
PRÁCTICA 3ª: CODE/DECODE BUG. LÍNEA DE COMANDOS.
-PREGUNTAS Y DUDAS.



NÚMERO 4:

- CREA TU SEGUNDO TROYANO, INDETECTABLE E INMUNE A LOS ANTIVIRUS.
CONOCIENDO EL RADMIN.
GESTIONANDO UNA SALA DE ORDENADORES.
OCULTANDO EL RADMIN.
INSTALANDO EL RADMIN EN EQUIPOS REMOTOS.
- OCULTACIÓN DE IP POR NOMBRE DE DOMINIO.
- CREA LETRAS DE IMPACTO PARA TUS DOCUMENTOS (LETRAS DE FUEGO).
- CONSIGUE UNA IP FIJA.



NÚMERO 5:

-HACK-OPINION: LA PIRATERÍA EN INTERNET.
-ROOTKITS: LA PESADILLA DE CUALQUIER ADMINISTRADOR.
-ROOTKITS: EL SR. NTROOT.
-WAREZ: APPZ, GAMEZ, MP3Z, DIVX, FTPZ, 0-DAY.
-APRENDIENDO A COMPILAR PROGRAMAS. COMPILA TU PROPIO NETCAT.
-BUGS, ERRORES Y OTRAS FORMA DE JOD...
-NETBIOS: ESTUDIO Y PENETRACIÓN DE SISTEMAS.
-ASESINADOS POR LA LSSI.
-LISTADO DE ORDENES PARA NETBIOS.
-HACK-OPINION: PAGOS POR INTERNET SEGUROS YÁ.



NÚMERO 6:

- PASA TUS PELICULAS A DIVX (STREAMING)
- PASA TUS PELICULAS A DIVX II (CODEC DIVX)
- PUERTOS & SERVICIOS
- eMule: EL NUEVO REY DEL P2P
- NUEVA SECCION: PROGRAMACION DESDE 0
- CURSO DE VISUAL BASIC
- IPHXC: EL TERCER TROYANO DE HXC
- TENDENCIAS ACTUALES EN CODIGO MALICIOSO
- OCULTACION DE FICHEROS. METODO STREAM (ads)
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 7:
- PROTOCOLOS: POP3
- PASA TUS PELICULAS A DIVX III (EL AUDIO)
- PASA TUS PELICULAS A DIVX IV (MULTIPLEXADO)
- CURSO DE VISUAL BASIC: LA CALCULADORA
- IPHXC: EL TERCER TROYANO DE HXC II
- APACHE: UN SERVIDOR WEB EN NUESTRO PC
- CCPROXY: IV TROYANO DE PC PASO A PASO
- TRASTEANDO CON EL HARDWARE DE UNA LAN



NÚMERO 8:

- CURSO DE LINUX
- APACHE: COMPARTE ARCHIVOS
- REVERSE SHELL
- CURSO DE VISUAL BASIC: MAS CALCULADORA
- PROTOCOLOS Y SU SEGURIDAD: SMTP



NÚMERO 9:

- CURSO DE LINUX (Sistema de archivos)
- APACHE: COMPARTE ARCHIVOS MEDIANTE WEB.
- CURSO DE VISUAL BASIC: MI 1ª DLL \ ACCESO A DATOS
- PORT SCANING: NMAP
- SERIE RAW: IRC



NÚMERO 10:

- CURSO DE LINUX (Gestión de usuários)
- APACHE + MySQL + PHP=Trio de Ases
- CURSO DE VISUAL BASIC: ACCESO A DATOS(II)
- XML: El futuro de la trasferencia de datos
- SERIE RAW: DCC



NÚMERO 11:

- Curso de linux: programacion
- Visual Basic: IIS bug exploit
- Apache como proxy
- Serie Raw: FTP
- Validacion XML: DTD
- Historia: Lady Augusta Ada Byron



NÚMERO 12:

- Curso de linux: programacion C.
- Visual Basic: IIS bug exploit. Nuestro primer Scanner.
- APACHE: Configuralo de forma segura.
- Serie Raw: FTP(II)
- VALIDACION XML: DTD (II)



NÚMERO 13:

- Curso de linux: programacion C(II).
- Visual Basic:Nuestro primer proyecto.
- APACHE: Configuralo de forma segura.
- Serie Raw: HTTP.
- CURSO XML: DOM.

CONSIGUE LOS NUMEROS ATRASADOS EN:

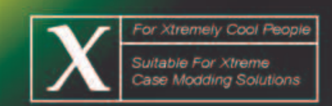




"Be Cool! Be Aerocool!"



RAFAEL ALBERTI, 24 BAJO
01010 VITORIA-GASTEIZ (SPAIN)
TEL.: 902-227733 / 945-176647
FAX.: 94-4342433
http://www.biomag.biz
E-MAIL: compras@biomag.biz







¡La nueva generación de cajas de ordenador!



¿Tiene un ordenador siempre que parecer un ordenador?
¡¡ NO !! ¡¡ NO MAS MONOTONIA!!!

¡¡Ahora, transforma tu ordenador "cuadrado" en cualquier cosa!! Presentamos las nuevas cajas "LUBIC". Gracias al concepto modular de "LUBIC", dispones de todo lo necesario para crear tu propio diseño de caja de ordenador con los elementos de aluminio incluidos en el producto

Planifica cuidadosamente el plan de construcción y los materiales para transformar tu ordenador en algo más que una simple y aburrida caja de metal.
¡¡Todo es posible con las cajas "LUBIC" !! Puedes crear una caja "Escorpión", una caja "Elefante", una caja "bombardero B-52" o cualquier cosa que imagines. También puedes crear cosas que no sean ordenadores con los modulos "LUBIC" como portaretratos, expositores o mesas.

¿Es el cielo el límite?
Con los módulos "LUBIC", ¡¡NO EXISTEN LIMITES!!!
¡¡¡Da forma a todas sus ideas!!! ¡¡Las creaciones solo estan limitadas por tu imaginación!!!

Los paquetes estandar son:

1. LUBIC-3519-BL
2. LUBIC-3519-BK
3. LUBIC-3519-SV
4. LUBIC-4480-BL
5. LUBIC-4480-BK
6. LUBIC-4480-SV
7. LUBIC-AIRPLANE-BL
8. LUBIC-AIRPLANE-BK
9. LUBIC-AIRPLANE-SV





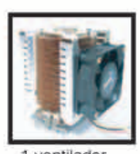


Serie Deep Impact

DP-102




Maximo Diseño


1 ventilador


2 ventiladores

Caracteristicas

1. Alta conductividad térmica
2. Totalmente fabricado en cobre con un tubo superconductor para un maximo rendimiento
3. Super Diseño circular
4. DP-102 ofrece una solución de doble ventilador - pueden usarse uno o dos ventiladores. (Ventiladores no incluidos en el paquete)
5. Incluye dos sets de adaptadores de ventilador de 70mm a 80mm.
6. El disipador puede ser rotado 360 grados y los ventiladores pueden ser instalados en cualquier posición.
7. DP-102 puede ser "Universal" compatible con AMD e Intel P4

Aplicaciones

AMD: Athlon XP 3600+ y superiores
Intel: P4 Socket 478 3.6Ghz y superiores (solo aplicable a la versión "Universal", no disponible para la versión "AMD")

Disipador

Dimensiones =Tubo -100 mm, 36+ láminas - dia. 66mm
Material =Tubo Superconductor + Láminas de Cobre